LVEE 2007

Linux Vacation / Eastern Europe

Тезисы конференции

14-17 июня 2007 г.

Оглавление

Опыт успешной разработки коммерческого проекта на базе решений OpenSource

Андрей Шевченко - ООО АСПЛинукс Плюс - Донецк, Украина

Рассмотрены особенности создания и внедрения проприетарного программного продукта, построенного с использованием ПО с открытым исходным кодом, на примере Acronis LiveMedia — встраиваемой мультимедийной системы, позволяющей использовать стандарнтный персональный компьютер в качестве полнофункционального развлекательного центра.

введение

На конференциях часто обсуждается использование OpenSource для разработки снова-таки OpenSource решений. Однако, давайте обратим внимание и на интересную нишу, сочетающую в себе с одной стороны решения с открытыми исходными текстами, а с другой - коммерческие продукты.

мотивация использования OpenSource

Как и для любого коммерческого проекта, выбор базы для строительства опирается на несколько вариантов. Каждый из вариантов обладает своими преимуществами и недостатками. Рассмотрим некоторые из них.

разработка с нуля

Преимущества такой разработки очевидны. Во-первых, разработчики чётко владеют ситуацией по архитектуре и интерфейсам всех компонент системы. Весь проект в целом развивается по однородному графику. Во-вторых, выбор инструментария зависит не от конкретных реализаций каких-то частей системы, а лишь от самой задачи и результата общения между менеджером проекта и разработчиками. Но при этих, казалось бы весомых, преимуществах, существует ряд недостатков. Первый из них - это безусловно скорость разработки до первого выпуска продукта на волю. Второй, проистекающий из первого - высокая стоимость вложений на начальном этапе, а следовательно, повышение стоимости результата. И самый ощутимый третий - скорость реакции и её стоимость на замеченные проблемы (их устранение) или на запрашиваемые расширения функциональности.

использование закрытых программных продуктов

При разработке тем не менее можно воспользоваться продуктами третьих фирм, помогающими решать данную задачу. С одной стороны это упростит разработку, ускорит её темпы, и даже не собьёт однородность развития, так как изменения интерфейса в оплаченном продукте могут серьёзно сказаться на имидже компании-разработчика. Но с другой - увеличение стоимости продукта возникнет за счёт лицензионных отчислений третьим фирмам. А неожиданное прекращение существования фирмы-производителя или самого продукта может вывести из строя проект на некоторое время и повлечь за собой дополнительные затраты.

использование ПО с открытыми исходными кодами

Можно выделить следующие преимущества использования продукции третьих фирм, но с политикой лицензирования на основе открытых исходных кодов:

- низкая стоимость вложения в разработку базы;
- независимость от действий со стороны разработчика.

Но тем не менее, существуют и недостатки такого подхода. Так, например, наблюдается неравномерный процесс работы над различными составляющими системы. В частности, нам пришлось столкнуться с серьёзными изменениями, связанными с изменением некоторых интерфейсов подсистемы HAL; переход от версии PyGame 1.6.x к 1.7.x привнёс некоторые изменения API, и так далее.

окончательный выбор

Учитывая изложенные выше недостатки и преимущества, мы пришли к использованию открытого ПО для проекта. Во-первых, базируясь на дистрибутиве, продукт получил обкатку и, с другой стороны, продвижение самого дистрибутива. Во-вторых, значительно уменьшена стоимость вложений в разработку, в том числе увеличена её скорость. В-третьих, внимание разработчиков в большей степени уделено проекту, нежели его окружению.

обратная связь

Всегда найдётся человек, который, услышав о закрытых исходниках, будет всячески критиковать такой проект. Для того, чтобы предупредить подобную критику в этой части коснёмся ответов на вопросы: "А что продукт дал обществу свободного ПО? Какая же обратная связь и польза?" Поскольку мультимедийный проект направлен в основном на работу с медиа-устройствами, то чаще всего результатом обратной связи является поддержка или улучшение функционирования таких устройств. Так, в частности, благодаря нашей команде в проект ALSA было выслано более десяти заплаток для поддержки и корректирования работы драйверов с некоторыми кодеками и звуковыми платами. Дополнительно, в сердце системы, в ядро Linux были добавлены некоторые опции, корректирующие работу ACPI подсистемы. Проект v4l получил также поддержку некоторых тюнеров благодаря проекту.

сумма сказанного

Стоимость владения и вложенные средства в разработку минимизированы из-за использования базы в виде OpenSource. Невзирая на неоднородность развития некоторых подсистем в целом, скорость разработки и поддержки системы up-to-date не пострадала. В тоже время коммерческие проекты - это не всегда зло для OpenSource. В зависимости от области соприкосновения коммерческого и открытого ПО могут возникать различные способы взаимовыгоды.

источники

1. Acronis LiveMedia // http://www.acronis.ru/oem/products/livemedia

2. ASPLinux LiveMedia Edition // http://www.asplinux.ru/homeoffice/products/lme

Сервисы Novell для ОС Linux

Роман Гаранин - Брестский государственный технический университет - Брест, Беларусь

Сервисы Novell - эта целая группа коммерческих и открытых решений, работающих на различных платформах. ОС Novell Open Enterprise Server представляет собой попытку совместного использования ОС Linux и ряда технологий, выросших из платформы Novell NetWare. К фирменным Novell'овским сервисам относятся служба каталогов eDirectory, средства групповой работы GroupWise, средства прозрачной сетевой печати и работы с домашними каталогами, средства управления ZENworks, система хранения данных Cluster Services.

eDirectory

Служба каталогов Novell eDirectory (ранее Novell Directory Services) - иерархическая, объектно-ориентированная база данных, которая представляет все ресурсы организации в виде логического дерева. Ресурсами могут быть сотрудники, должности, серверы, рабочие станции, приложения, принтеры, службы, группы и т. д. Эффективное управление доступом к ресурсам достигается за счёт использования динамического наследования прав и механизмов эквивалентности по правам. Доступ к данным eDirectory можно получить через LDAP, XML, DSML, SOAP, OBDC, JDBC, JNDI, EJB, ActiveX и ADSI.

GroupWise

Novell GroupWise - комплексное решение для коллективной работы, предоставляющее возможность защищенного взаимодействия, интеграцию с антивирусным ПО и средствами защиты от спама. К основным возможностям GroupWise относятся:

- электронная почта с отслеживанием состояния и возможностью отзыва;
- основанная на правилах система управления сообщениями;
- мгновенные сообщения;
- проверка орфографии;
- управление заданиями, контактами и документами;
- синхронизация с PDA;
- поддержка кластеризации;
- поддержка GroupWise серверов NetWare, Windows и Linux (как SUSE LINUX Enterprise Server, так и Red Hat).

iPrint, iFolder, BorderManager, Identity Manage, инструмент iManager

Novell iFolder обеспечивает удаленный доступ к личным файлам. У каждого пользователя имеется папка Novell iFolder, доступная из любого места - с настольного компьютера, отсоединенного от сети ноутбука, сотового телефона, карманного ПК или из Интернет-кафе. Работа с файлами ведется локально, а по мере модификации файлов выполняется автоматическая синхронизация по сети, благодаря чему в любой момент с любого компьютера доступна последняя версия файла. Если пользователь создает или изменяет файл на компьютере, не подключенном к сети, обновления производятся при следующем соединении. Скорость доступа достаточно высока, так как Novell iFolder передает только изменения.

iPrint обеспечивает удаленную печать документов.

Имея iPrint, пользователи могут печатать на любом принтере, разрешенном им для

использования, вне зависимости от его местоположения - даже за файрволом. iPrint позволяет находить нужный принтер, при необходимости автоматически получать нужный драйвер и защищенно отправлять задание на печать. iPrint полностью совместим с IPP (Internet Printing Protocol), стандартным протоколом, определяющим операции и атрибуты для печати через Интернет. С помощью iPrint можно преобразовать в принтер IPP любой принтер, способный работать с сервером, вместо того, чтобы приобретать дорогостоящие IPP-принтера. iPrint автоматически создает Web-страницу, которая содержит все принтеры, принадлежащие данному серверу, и по мере необходимости устанавливает на рабочие станции необходимые компоненты.

BorderManager включает основные технологии файрвола и виртуальной частной сети, позволяет выполнять мониторинг деятельности пользователей в Интернете и управлять удаленным доступом к корпоративным ресурсам. Кроме того, BorderManager предоставляет управление Интернет-доступом и поддерживает многочисленные решения по фильтрации содержимого. Identity Manager автоматически синхронизирует учетные записи пользователей между разнородными службами каталогов и базами данных. Среди прочего оддерживаются Novell eDirectory, SAP R/3, SAP HR, каталоги LDAP, Linux, UNIX, Novell GroupWise.

iManager - общий для eDirectory и Identity Manager веб-инструментарий управления содержимым каталога. Ядро iManager базируется на технологии сервлетов и использует в сервер приложений Tomcat и веб-сервер Apache.

ZENworks Suite

Novell ZENworks Suite представляет собой комплексное решение для управления ИТ-ресурсами на основе электронных персон. В список решаемых задач входит:

- управление рабочими станциями, серверами, карманными ПК;
- управление Linux;
- управление исправлениями, компоновка программного обеспечения;
- управление данными, перенос персональных данных.

В средствах управления Linux (ZENworks Linux Management) используется технология автоматизации на базе политик для установки, управления и эксплуатации Linux-ресурсов на протяжении всего жизненного цикла Linux-систем с возможностями изоляции рабочих станций, загрузки образов, дистанционного управления, инвентаризации и управления программным обеспечением. В состав системы входит веб-консоль администрирования ZENworks Control Center.

ZENworks автоматически проверяет зависимости между пакетами программного обеспечения и сообщает администраторам о конфликтах до завершения установки. Возможность отката позволяет перейти к предыдущей конфигурации установленного ПО или к состоянию на определенную дату.

Включенный в SUSE Linux, ZENworks Linux Management Client написан на C# (Моно), поддерживает репозитории capt, open carpet, Red Carpet и ZENworks Linux Management. Клиент запускается как системный демон, имеет модульную архитектуру и использует как командную строку, так и графический пользовательский интерфейс.

Novell Cluster Services

Novell Cluster Services обеспечивает управление всеми ресурсами сети хранения данных (Storage Area Network – SAN). Novell Cluster Services интегрирован с Open Enterprise Server и вместе эти продукты обеспечивают свободный непрерывный доступ к данным и ресурсам, а также масштабирование систем хранения. С помощью Novell Cluster Services можно объединить несколько серверов в кластер высокой готовности. В случае сбоя в работе какого-либо сервера, тома хранения, приложения, веб-сайты и другие ресурсы можно сконфигурировать соответствующим образом и за считанные секунды автоматически

перевести их на другие серверы кластера.

NetWare, SuSE Linux Enterprise Server, openSuSE

В данный момент Novell продвигает операционные системы NetWare, продукты, базирующиеся на SUSE Linux, а также спонсирует полностью открытый проект openSuSE.

С приобретением в 2003 г. компании SuSE Linux Novell начала разработку целого ряда собственных дистрибутивов Linux, а также ряда Open Source проектов, включая поддержку системы виртуализации XEN и платформы Mono, а также собственные проекты Open Source, наиболее известные из которых - система безопасности Novell AppArmor (аналог SELinux), Hula (серверное ПО для коллективной работы, включающее почту и календарь) и Bandit (ПО для управления электронными удостоверениями).

Кроссплатформенная система для установки и управления операционными системами в распределенных гетерогенных сетях

Павел Чеботарев, Денис Пынькин, Игорь Ильенко - Минск, Беларусь

Рассматриваются общие принципы работы системы, ее структура и концепция применения для установки и управления операционными системами

Гетерогенность на сегодняшний день является неотъемлемым свойством многих информационных систем. При совмещении разнородных систем важное значение имеет производительность, надежность, совместимость, управляемость, защищенность, расширяемость и масштабируемость комплексной гетерогенной системы. Попытка обеспечить указанные качества реализована в единой системе управления ClusterIM - кроссплатформенной микроядерной системе для установки и управления операционными системами в распределенных гетерогенных сетях. Система реализована на объектно-ориентированном языке Ruby.

Особенности системы ClusterIM позволяют ее применять в различных сферах деятельности, таких как:

- виртуальный хостинг;
- управление компьютерным парком предприятия;
- установка и управление вычислительным кластером;
- массовая установка операционных систем для OEM производителей;
- локальная установка операционной системы.

ClusterIM может использоваться как для сетевых решений, так и в локальных целях. Основные возможности рассматриваемой системы включают:

- автоматизация установки операционной системы;
- простая интеграция поддержки операционной системы;
- расширение функциональности системы с помощью модулей OEM производителей;
- централизованное управление узлами в гетерогенных сетях;
- простая интеграция с существующей инфраструктурой предприятия.

Изначально администрирование сети требует применения специализированных средств управления в рамках каждой платформы, что затрудняет централизацию процесса управления. Возникает необходимость в средствах комплексного управления гетерогенной сетью, не зависящих от протокола, масштабируемых, обладающих единой консолью управления. Перечислим основные особенности системы, позволяющие управленять узлами в сети:

- установка операционной системой - может быть локальной (с таких носителей, как CD, DVD, USB-flash) либо удаленной (PXE, Etherboot);
- создание виртуальных узлов (включая создание виртуальной машины и установку на нее операционной системы);
- управление состоянием операционной системы за счет поддержки таких операций, как старт, стоп, бэкап и миграция;
- управление ресурсами операционной системы (предусматривается управление сервисами операционной системы, а также установка, обновление и удаление

программного обеспечения).

Каждый программный элемент обладает своим собственным, изолированным от других элементов, окружением. Появляется возможность использовать одновременно множество операционных систем и при этом значительно повышается безопасность системы в целом.

При использовании такого рода распределенных виртуальных систем, принципов терминального доступа и интеллектуального управления агентами, мы можем получить гетерогенную вычислительную виртуальную среду, максимально использующую все ресурсы физических машин, входящих в систему.

источники

1. Гетерогенные сети и устройство. Режим доступа. http://skif.bas-net.by/bsuir/mpich_userguide_site/node16.html

2. А. А. Николаев. Комплексные решения по управлению информационной средой предприятия. http://www.ccc.ru/magazine/depot/99_02/read.html?0202.htm

3. Д.А. Пынькин, П.В. Чеботарев, Р.Х. Садыхов. Концепция применения виртуальных машин в кластерных распределённых системах.// Труды международной конференции "Информационные системы и технологии", Минск, 2006

Использование операционной системы Linux в активных сетевых устройствах SOHO сегмента

Сергей Довнар - Team Leader группы Messaging solutions, СП ООО "МЦС" - Минск, Беларусь

введение

В последнее время широкое распространение получили активные сетевые устройства, совмещающие в себе сразу несколько функций. Как пример, можно привести маршрутизатор Linksys WAG200G. В этом устройстве совмещены ADSL модем, Wi-Fi точка доступа и коммутатор на четыре порта 10/100 Mbit. В качестве операционной системы используется Linux с ядром ветки 2.4 и проприетарными драйверами для Wi-Fi и ADSL частей. Возникает резонный вопрос: возможно ли использование Linux и в других подобных устройствах, несмотря на то, что производитель такую возможность не предусмотрел?

аппаратная часть

Для того, чтобы выяснить, можно ли запустить ядро Linux на устройстве, необходимо детально изучить объект, над которым будут проводиться эксперименты. В моём случае таким объектом оказалась Wi-Fi точка доступа D-Link DWL-2100AP. Во время обновления прошивки пропало питание, и устройство перестало выполнять свои функции. Доступ к консоли у устройства не предусмотрен, а сетевое соединение установить было невозможно.

В первую очередь была снята крышка и изучена находившаяся внутри корпуса плата. Недолгий поиск в Интернет позволил обнаружить несколько форумов, содержащих полезную информацию, в результате изучения которых были установлены следующие технические характеристики устройства:

- Платформа: ar531x (Atheros)
- RAM: 16 Mb
- Flash ROM: 4 Mb
- Тип процессора: MIPS.

На странице проекта OpenWRT была найдена информация по портам RS-232 и JTAG, распайка под которые есть на плате устройства. RS232 используется как системная консоль, JTAG – это технологический разъём, используемый для перепрошивки памяти устройства.

Единственный способ подключиться к устройству для дальнейших экспериментов - использование RS-232 порта. Однако производитель не пожелал предоставить кому-то возможность такого доступа к устройству, и порт оказался не разведён до конца. Доработка свелась к подключению схемы преобразователя на микросхеме MAX232 для получения полнофункционального RS-232. Для этой цели можно использовать схему из технического описания микросхемы.

Далее был запущен minicom и настроены следующие параметры обмена: 9600 8N1. Затем были соединены кабелем порты устройства и компьютера и подключено питание к точке доступа. При правильном выполнении описанных выше процедур на экране должны появиться строчки, выдаваемые загрузчиком.

В случае описываемого устройства текст был следующим:

```
ar531x rev 0x00005850 firmware startup...
SDRAM TEST...PASSED
```

```
WAP-G02A Boot Procedure V1.0
-----------------------------------------------------
Start ..Boot.B12..
Atheros AR5001AP default version 3.0.0.43A

0

auto-booting... Attaching to TFFS... done.
```

Процесс загрузки прерывается нажатием Esc, после чего можно изучить возможности, которые предоставляет загрузчик. Вот какие параметры установлены в системе по-умолчанию:

```
[Boot]: p
boot device : tffs:
unit number : 0
processor number : 0
file name : /fl/APIMG1
inet on ethernet (e) : 192.168.1.50:0xffffff00
flags (f) : 0x0
other (o) : ae
```

Итак, теперь известно, где лежит образ, который содержит ПО, необходимое для работы точки. Внимательное прочтение помощи по командам загрузчика показало, что он умеет загружаться по сети, используя TFTP.

После подготовки TFTP-сервера и выкладывания на него ядра, взятого на OpenWRT, параметры загрузки необходимо изменить так, чтобы устройство начало загружаться по сети. В рассматриваемом случае изменения имели следующий вид:

```
$ae(1,0)tftpserver:vmlinux h=172.16.255.1 e=172.16.255.253:0xffffff00 f=0x80
```

Попытки устройства загрузиться по сети выглядят следующим образом:

```
Attached TCP/IP interface to ae1.
Attaching network interface lo0... done.
Loading...
```

Далее идёт стандартный вывод загружающегося ядра Linux, в конце которого оно впадает в панику, не найдя корневую файловую систему. Это совершенно естественно, поскольку на TFTP находится только ядро и образа файловой системы там нет. Данный этап имеет своей целью выяснить возможность ядра без проблем загрузиться на предложенном ему оборудовании.

Проделанные с устройством эксперименты продемонстрировали, что:

- с устройством можно взаимодействовать посредством интерфейса RS-232;
- устройство может загружаться «извне» - по сети;
- платформа устройства позволяет использовать Linux после некоторых доработок.

программная часть

Можно выделить четыре части ПО, которые необходимы для работы устройства:

- загрузчик;
- ядро операционной системы;
- драйвера аппаратной части устройства;
- системное ПО.

На начальном этапе был использован загрузчик VxWorks, встроенный в систему. Для остальных частей возможна загрузка по сети. Необходимо создать образ RAM-диска, на котором будет расположено системное ПО. Сделать это можно, скачав исходный код следующим образом:

1. Создать в домашнем каталоге подкаталог (например – dwl-2100ap) и перейти в него.

2. Ввести команды:

```
svn co https://svn.openwrt.org/openwrt/trunk/
svn co https:// svn.openwrt.org/openwrt/packages ./trunk/feeds/packages/
cd ./trunk/
make package/symlinks
make menuconfig
```

По окончании конфигурирования будущей системы (при этом необходимо не забыть «встроить» образ файловой системы в ядро) выполняется команда make world. Полученный образ выкладывается на TFTP-сервер, после чего загружается в устройство по сети. Если все приведенные операции выполнены правильно, результатом оказывается работающее устройство, обладающее гораздо большим, по сравнению с первоначальным, функционалом.

заключение

Приведенные инструкции следует рассматривать не как прямое руководство к действию, а только как указание направления, в котором можно двигаться. Автор не несёт ответственности за возможные повреждения устройств, над которыми будут проводиться эксперименты.

Описанная схема исследования работает для всех устройств, но поддержка конкретной модели проектом OpenWRT – вопрос личной удачи экспериментатора. В затронутой теме существует огромное поле для экспериментов. Получив знания об аппаратной части любого устройства и проявив творческий подход, всегда можно добиться того, что не захотел или поленился сделать его производитель.

Проблемы защиты интеллектуальной собственности в сфере программного обеспечения

Илья Нечай - Луганск, Украина

Сейчас активно идёт процесс интеграции с мировым сообществом, это неоспоримо. Однако для интеграции на равных первым делом необходимо разработать систему законодательства. Для нас особенно интересна та его часть, которая регулирует вопросы интеллектуальной собственности. И это вполне естественно, ведь необходимо не только создать какой-то программный продукт, но и грамотно защитить право на него. То, о чём я расскажу, в принципе подходит не только для Украины, но и для большинства стран СНГ, ибо основа законодательства была едина, да и процессы развития во многом сходны.

Правительство Украины делает все возможное для вступления во Всемирную торговую организацию (ВТО) и интеграции в Европейский союз (ЕС). С 2001 года совершенствуются правовые основы в сфере интеллектуальной собственности. В 2001-2002 годах принят Криминальный кодекс Украины и внесены изменения в Кодекс Украины об административных правонарушениях. В них включены нормы, направленные на усиление криминальной и административной ответственности за нарушение прав интеллектуальной собственности. В 2002-2003 годах Верховной Радой Украины принят Гражданский кодекс, содержащий отдельную книгу “Право интеллектуальной собственности”, а также Хозяйственный и Таможенный кодексы Украины. После подписания в мае 2000 года Украино-американской совместной программы действий по борьбе с нелегальным производством оптических носителей информации это направление деятельности приобрело признаки государственной политики. Принятие в сжатые сроки соответствующих законов Украины, постановлений Кабинета Министров и других нормативно-правовых актов дало возможность сформировать правовое поле и ввести действенные механизмы реализации правовых норм по усилению ответственности за нарушение прав интеллектуальной собственности и недопущению производства и распространения контрафактной продукции. Также с целью обеспечения контроля за соблюдением норм законодательства.

Относительно программного обеспечения распоряжением Кабинета Министров Украины в мае 2002 года утверждена Концепция легализации программного обеспечения и борьбы с нелегальным его использованием. Концепцией предусмотрены: легализация программного обеспечения; организация борьбы с его незаконным использованием; развитие отечественной индустрии программного обеспечения. Ныне активно осуществляются мероприятия по реализации положений Концепции. Определимся, что такое программное обеспечение.

Термин "программа" произошел от греческого "programme" и первоначально определялся как содержание или план какой-либо деятельности, работ. С появлением ЭВМ под "программой" в прикладной математике стали понимать описание алгоритма (иначе говоря, последовательности) решения определенной задачи. К ЭВМ относятся не только технические средства, охраняемые традиционно патентным правом, но и специфичный по своей природе объект, являющийся результатом творческой деятельности человека. Программа для ЭВМ является органическим элементом и направлена на обеспечение ее работы, без программы ЭВМ - мертвая груда деталей, "hardware". При применении термина "программный продукт" подчеркивается статический признак, завершение какого-либо действия. Тогда как в термине "обеспечение" больше выражается функциональный признак определения, направленность на достижение какой-либо цели. Подводя итог, можно определить объект интеллектуальной собственности как допускаемый правом объективированный результат интеллектуальной деятельности, имеющий конкретного автора и отражающий его индивидуальность, что не

исключает, а напротив, предполагает реальную возможность воспроизведения его третьими лицами.

С тем, что такое программное обеспечение, мы определились. Теперь надо разобраться, а что же такое интеллектуальная собственность и почему её необходимо защищать?

Начнём с того, что само понятие "интеллектуальная собственность" не тождественно понятию собственности, оно исторически от него производно. На категории собственности покоится воспроизводство человеческой жизни, так как основные потребности общество удовлетворяет за счет собственности. Изначально особенность объектов интеллектуальной собственности, в отличие от других объектов гражданских правоотношений, заключается в свойстве нематериальности. Нематериальная природа результата интеллектуального труда не только ограничивает правомочие владения, но вместе с тем позволяет одновременно использовать его в разных местах и разными лицами, не создавая этому помех и ограничений.

Особые общественные отношения, складывающиеся по поводу объекта интеллектуальной собственности, являются отдельным объектом правового регулирования. Ранее подобные объекты исключались из числа охраняемых правом, но затем с развитием общественных потребностей появилась необходимость признания таких отношений правом, при этом преимущественной, органичной формой опосредования потребностей общества выступают дозволения, как наиболее эффективный и цивилизованный способ правового регулирования.

Для защиты своих прав на программу необходимо её защитить. Тут мы плавно переходим к вопросу лицензирования программного обеспечения.

Сам термин «Лицензия» в законодательстве используется в двух значениях:

1.Лицензия – разрешение компетентного государственного органа на осуществление определенного вида деятельности (из числа видов деятельности, подлежащих обязательному лицензированию);

2.Лицензия – разрешение обладателя исключительных прав на объект интеллектуальной собственности (художественное произведение, программу для ЭВМ, изобретение, товарный знак) использовать этот объект определенным образом.

По договору о предоставлении исключительных прав на программу для ЭВМ или базу данных (лицензионному договору) правообладатель передает другому лицу – пользователю – право использовать данный объект оговоренным с соглашении образом. Объем предоставляемых правомочий может быть различным и определяется соглашением сторон. Единственное ограничение по объему передаваемых прав вполне традиционно: правообладатель не может передать больше прав, чем имеет сам. Последнее особенно актуально, если права на объект интеллектуальной собственности возникли у правообладателя не непосредственно (в силу создания программы или базы данных), а были получены им по договору или по иным основаниям. Программное обеспечение может распространяться на условии открытого кода (OpenWare), или же без такого условия.

Ещё на заре движения за открытое программное обеспечение, его основатель Ричард Столлман сформулировал понятие свободное программное обеспечение, в котором отразились принципы открытой разработки программ в научном сообществе, сложившемся в американских университетах в 1970-е годы. Столлман явно сформулировал эти принципы, они же — критерии свободного программного обеспечения. Эти критерии оговаривают те права, которые автор свободной программы передаёт любому пользователю.

Программу можно использовать с любой целью («нулевая свобода»).

Можно изучать, как программа работает и адаптировать её для своих целей («первая свобода»). Условием этого является доступность исходного текста программы.

Можно распространять копии программы — в помощь товарищу («вторая свобода»).

Программу можно улучшать и публиковать свою улучшенную версию, с тем чтобы принести пользу всему сообществу («третья свобода»). Условием этого является доступность исходного текста программы.

При распространении программного обеспечения на условиях открытого кода правообладатель, предоставляя пользователю право использования программного обеспечения, передает также исходные коды программы. При этом, как правило, пользователю предоставляется также право модифицировать исходные тексты, перерабатывая и совершенствуя их.

Дальнейшее использование полученных в результате такой переработки программных продуктов различается в зависимости от вида лицензии. В настоящее время практикой выработано два «семейства» типовых лицензий на передачу программного обеспечения с открытым кодом: GNU GPL и FreeBSD.

Основное их различие заключается в «наследуемости» свойства открытого кода: согласно условиям лицензии GNU GPL, все программные продукты, полученные в результате переработки или модернизации распространяемого на таких условиях программного кода, также могут распространяться далее только на условиях GNU GPL. Это, с одной стороны, способствует прогрессу в развитии программного обеспечения, с другой – нарушает имущественные интересы некоторых разработчиков, вложивших серьезные средства в модернизацию программного кода.

В течение последних лет мы наблюдаем очень быстрое развитие систем передачи данных и всемирной компьютерной сети интернет. С одной стороны Интернет значительно упрощает процесс создания программ, так как даёт практически неограниченные возможности по совместной разработке проектов. Авторы могут жить в разных частях света, говорить на разных языках и никогда не видеть друг друга, но тем не менее совместно разрабатывать программу. Но это явление имеет и обратную сторону. Интернет, как удобное и простое средство передачи информации содействует многочисленным нарушениям авторских прав. Самые распространённые среди них – это нарушения личных неимущественных прав автора, а именно: на авторство, на имя, на обнародование, на уважение репутации. Примеров такого рода огромное множество. Это и так называемые «варезные» сайты на которых размещаются программы со взломанной защитой, и сборники серийных номеров для различных программ, и различные файлообменные сети, например eMule. Но привлечь к ответственности личность, которая распространяет программное обеспечение с нарушением прав его авторов, практически невозможно. Тут очень важно установить меру ответственности провайдеров и владельцев такого рода сайтов. В США, например, владелец сайта признаётся нарушителем, если ему известно, что программа защищена авторским правом, а он использует её без дозволения автора. В теории это звучит хорошо, но необходимо сказать, что большинство авторов сайтов разрабатывают условия, в соответствии с которыми вся ответственность переводится на пользователей, а случаев привлечения к ответственности владельцев сайтов единицы.

Ни для кого не секрет, что у нас практически в любом ларьке можно купить пиратский диск с программами, лицензии на которые стоят тысячи долларов. Изготовление пиратских копий программ нарушает вещное право право автора, а именно исключительное право на использование произведения. Предотвращением подобных нарушений должен заниматься специально созданный орган. Таким образом, нарушение прав на объекты интеллектуальной собственности можно отнести к двум типам: во-первых, связанные с использованием способов индивидуализации (как правило товарных знаков) и, во-вторых, пиратство.

По форме защиту можно разделить на две категории: юрисдикционная, то есть та, что обеспечивается с использованием госаппарата (уголовное и гражданское судопроизводство и

производство по делам об административных нарушениях) и неюрисдикционная, то есть та, что охватывает собой действия граждан и организаций (в том числе через их представителей - патентных поверенных), осуществляемая ими самостоятельно, без обращения к госорганам.

Более действенными являются естественно юрисдикционные способы. А пока лишь скажем, что правильный выбор лицензии очень сильно поможет в защите программы. Я конечно склоняюсь к лицензии GPL, ведь ещё Бенжамин Франклин говорил о том, что если мы используем чужие изобретения, то должны предоставить свои для всеобщего блага.

список литературы

1. Григорьев И. «А ты установил лицензионное программное обеспечение!?» // Юридична газета, №20, 4 ноября 2004г., с. 20-22
2. Тезаурус научно-технических терминов/Под ред. Ю.И.Шемакина. - М.: Воениздат, 1972
3. Ожегов С.И., Шведова Н.Ю. Толковый словарь русского языка. - М: Русский язык, 1983
4. Ровный В.В. Проблемы объекта в гражданском праве: Учебное пособие. - Иркутск, 1998.
5. Алексеев С.С. Теория права. - М.: Издательство БЕК, 1995.
6. Лицензии на программное обеспечение: понятие и виды // http://www.parkmedia.ru/lib.asp?ob_no=570

LILALO - ПО ведения журналов работы в консоли UNIX-систем

Игорь Чубин - Учебный центр "Сетевые технологии" - Киев, Украина

LILALO позволяет автоматически фиксировать полный ход работы с терминалом Unix-системы, включая текст команд и результат их выполнения, текущий каталог вызова, время вызова и множество других. Кроме записи работы с терминалом автоматически фиксируются изменения, сделанные с помощью текстового редактора. Полученные в ходе записи данные могут быть представлены в формате XML, пригодном для дальнейшего анализа хода работы, создания заготовок документации и сценариев командного интерпретатора.

В процессе работы в консоли Unix/Linux-системы, будь то непосредственное выполнение задач администрирования, экспериментирование с целью найти и описать решение какой-то задачи или самообучения, демонстрация приёмов работы на живых примерах или что-то другое, часто возникает необходимость зафиксировать происходящий в консоли процесс.

Записи нужны для того, чтобы или просто использовать при попытке повторить те же действия, но в другой раз, или потом, доработав и снабдив необходимыми комментариями и ссылками, превратить их в полноценную документацию.

Запись обычно выполняется одним из нескольких способов:

- запись вручную на бумаге;
- запись вручную в электронном виде;
- запись путём копирования мышью в текстовый редактор;
- с применением программы script.

Каждый из этих способов имеет собственные недостатки.

Недостатки существующих способов записи:

запись вручную на бумаге

Преимущества:

- может использоваться для записи команд, которые выполняются на другом компьютере или демонстрируются с помощью проектора.

Недостатки:

- долго;
- неудобно;
- может содержать ошибки;
- непригодна к дальнейшей электронной обработке.

Запись вручную в электронном виде обладает теми же достоинствами и недостатками, что и в случае ручной записи, но результат записи поддаётся дальнейшей электронной обработке.

Запись путём копирования в текстовый редактор имеет то преимущество, что копирование выполняется быстро и без ошибок. Недостатком является то, что копирование в текстовый редактор требует дополнительных действий и, что особенно важно, при обучении -- оно невозможно во время демонстрации команд.

В Unix существует программа script, которая автоматически выполняет запись всего

происходящего на терминале, где она запущена.

Преимущества использования программы script для записи:

- запись производится прозрачно;
- может выполняться во время демонстрации;
- запись не содержит ошибок.

Недостатки:

- необходимость в обработке после завершения записи;
- запись может производиться только для действий выполняемых непосредственно в командной строке.

Предлагаемое решение, система ведения журналов работы с терминалом Unix-системы LiLaLo [1], свободно от всех перечисленных выше недостатков и обладает рядом преимуществ.

LiLaLo использует для записи программу script. Однако, в отличие от программы script в чистом виде, во время записи фиксируются не только команды и результат их работы, но и множество дополнительной информации о командах. Это позволяет в дальнейшем более полно реконструировать ход работы. Кроме того, информация, которую LiLaLo автоматически записывает при ведении журнала, позволяет выполнять анализ хода работы и автоматически создавать заготовки для сценариев командного интерпретатора.

Автоматическая запись дополнительной информации о командных строках возможна за счёт модификации приглашения командного интерпретатора. Хотя визуально это (практически) никак не заметно, приглашение командного интерпретатора модифицируется, и в него, в скрытом виде, добавляется несколько параметров, характеризующих команду, которая набирается в этом приглашении и будет выполнена. В их числе:

- текущий каталог, из которого производится вызов команды;
- время;
- код завершения предыдущей команды и ряд других.

Помимо того, что производится запись всего хода работы в командной строке, автоматически фиксируются все изменения в файлах, сделанные с помощью текстового редактора. Есть возможность делать скриншоты и показывать в журнале окна, которые, возможно, имеют непосредственное отношение к производимым в консоли действиям.

Записанные данные хранятся в формате программы script, то есть, фактически, непосредственно в виде набора команд терминалу. Они могут быть обработаны и представлены в структурированной форме, в виде XML-файла. Который в дальнейшем может быть либо преобразован в HTML-файл и визуализирован, либо может просто попасть в хранилище.

Анализ терминального скрипта, преобразование его в XML и визуализация при помощи веб-интерфейса выполняется в реальном времени и без всякого дополнительного участия пользователя.

источники

http://xgu.ru/wiki/LiLaLo -- домашняя страница проекта LiLaLo

SUN STUDIO PERFORMANCE ANALYSER

Алексей Лапшин - Sun Microsystems - Санкт-Петербург, Россия

Выполнен обзор возможностей Sun Studio Performance Analyser.

Рассмотрены анализ и оптимизация нативных(C/C++/Fortran) и Java-приложений на Solaris/OpenSolaris/Linux ОС.

Нередко перед разработчиком встает вопрос - "почему моя программа работает недостаточно быстро ?" В этот момент необходим эффективный способ избавиться от тормозящих факторов. Даже в несложной программе таких составляющих достаточно: неэффективный алгоритм, ожидание ресурсов, зависимость по данным и проч. В больших программных комплексах решение этих вопросов может превратиться в головную боль на долгое время. Ниже рассматривается использование Sun Studio Performance Analyser для поиска и анализа проблем производительности как нативных(C/C++/Fortran) так и Java - приложений, что позволяет сократить время разработки и повысить скорость работы приложения.

Sun Studio Performance Analyser предоставляет пользователю широкий спектр возможностей(GUI и CLI интерфейсы) для профилировки разных типов приложений. Также имеется возможность генерации множества представлений результата эксперимента.

Примеры профилировки программ на языке C демонстрируют преимущества Sun Studio Performance Analyser при поиске и анализе неэффективно написанного цикла. Простые преобразования исходного кода могут ускорить выполнение на порядок. Аналогичный результат может быть достигнут при использовании оптимальных ключей компиляции из командной строки.

Рассматривая те же примеры на java можно заметить, что одинаковые проблемы имеют одинаковые решения вне зависимости от исходного языка программирования. Sun Studio Performance Analyser позволяет исследовать как java byte-code, так и оптимизированные Java HotSpot VM функции.

Казалось бы, профилировка программы с использованием библиотеки OpenMP не должна ухудшать быстродействие приложения. Однако на практике OpenMP иногда приводит не к ускорению, а замедлению программы. Нами исследован ряд случаев на предмет способов поиска и решения проблем.

Использование Sun Studio Performance Analyser предоставляет также средства для профилирования функций Solaris kernel, Thread Analyzer (включая GUI и CLI интерфейсы) и другие возможности.

Автоматизация задач на серверах. Опыт поддержки однотипных *NIX серверов

Антон Линевич - Киев, Украина

Каждый системный инженер сталкивается с проблемой нехватки времени. Количество серверов становится все большим, а времени на их поддержку остается все меньше. Возникает вполне разумное желание автоматизировать часть работы по их обслуживанию, централизовать управление групп серверов. Реализовать это можно, используя написанные собственноручно решения и выделяя время на их поддержку, либо можно использовать готовые решения, имеющие нужную функциональность.

Свободное время хочется использовать по полной и как можно дальше от вычислительных центров. Именно в период отдыха зачастую приходят гениальные мысли и находятся ответы на долгожданные вопросы.

К сожалению, такое времяпровождение недоступно до тех пор, пока не будет обновлено программное обеспечение, не будет обеспечена исправная работа системы мониторинга и достоверность информации о состоянии систем и серверов. Обычно требуется также проверить, а не вышло ли долгожданное обновление ядра, которое, в случае его выхода, непременно нужно испробовать и внедрить на группу серверов, не забыв обновить изменившиеся файлы конфигурации и скопировать их на все сервера, где установлены соответствующие пакеты. Ну и конечно, в срочном порядке необходимо применить patch к текущей версии ОС на все сервера, дабы уберечься от грядущих сетевых атак.

Какое-то время автор делал все перечисленное "вручную", открывая терминал на каждый из серверов и выполняя цепочку одних и тех же команд на всех серверах по очереди. Желание повторять действия пропало после нескольких раз, но появились другие:

- освободить себя от рутинных задач, которые машина вполне может выполнять самостоятельно, а что не может - сделать .
- по возможности централизовать выполнение задач для комфортного управления и быстрого обновления систем.

Начиналось все с такого shell-скрипта:

```
for hosts in "server1 server2 server3"
  do ssh $host do_something
done
```

Какое-то время это выручало, но недостатки были видны сразу. Не устраивало следующее:

- медленно: каждый процесс начинает выполняться только по завершении предыдущего;
- вывод неудобен для вычитывания, поскольку нет явного указания, какая строка принадлежит выводу какого сервера;
- сложность обслуживания для большого количества групп серверов и более сложных задач/команд.

В качестве быстрого решения на свет появился небольшой скрипт, который позволял запускать команды параллельно на группе серверов. Имя ему было присвоено невзрачное, но весьма понятное: farm_cmd (http://a.linevich.com/files/farm_cmd). По мере того, как скрипт исправно выполнял свою работу, в голове его автора вырисовывались общие черты совершенно необходимого законченного программного решения. Требования:

- скорость работы: желательно выполнить все до того, как выйдет очередной стабильный релиз ядра, и при этом успеть вечером на свидание;
- открытый код: с одной стороны "религия" не позволяет использовать "черные коробки", но с другой, что еще делать в освободившееся время, не копаться в коде. Вдруг понадобиться дополнительная функциональность?
- простота инсталляции: слишком требовательные задачи не вдохновляют на творчество (видимо поэтому более крупные задачи разделяют на мелкие);
- простота в использовании: идеально, если все останется так как и было ;-)

Всем известная поисковая система привела к следующим готовым решениям:

- clusterit, http://www.garbled.net/clusterit.html - написан на C, скорее всего не развивается автором, выглядит как законченный проект, в основном для параллельного компилирования (это мнение этой автора статьи).
- Tentakel, http://tentakel.biskalar.de/ - написан на python, на время просмотра не был готов к использованию.
- OpenPBS (Open Portable Batch System), http://www.openpbs.org/about.html - перешли на коммерческую версию. Выглядит симпатично, есть даже веб-интерфейс.
- Capistrano, http://manuals.rubyonrails.com/read/book/17 - написан на Ruby, незнакомый язык программирования, сам проект весьма интересен, слабо развивается и поддерживается. Показался трудным в использовании.
- Sun Grid Engine, http://gridengine.sunsource.net/ - написан на C, требует наличия запущенного сервиса (демона) на клиентских машинах. Не подошел именно из-за удаленного сервиса. Хочется использовать имеющиеся средства: ssh, telnet, scp, ftp, и без установки дополнительных утилит на удаленных серверах.
- fanout, http://www.stearns.org/fanout/ - написан на BASH, на момент просмотра не уступал по функциональности farm_cmd, но при этом имел проблемы с обработкой вывода ошибок.
- mussh, http://sourceforge.net/projects/mussh/ - написан на BASH, не поддерживается с 2002 года.
- Cluster SSH - графическое приложение, неплохой выбор когда требуется перенаправить стандартный ввод на несколько терминалов. Не подходит из-за другой направленности.

И все-таки решение было принято в пользу Clusterit, как законченного решения, вполне работоспособного и проверенного "в бою". Пакет содержит следующие компоненты:

- dsh - выполнение команды на кластере (от англ. группа) машин параллельно;
- run - выполнение команды на случайной ноде;
- pcp - копирование файлов на группу машин;
- pdf - вывод команды df в общий вид на группе машин;
- prm - удаление файлов на группе машин.

Дополнительно, имеются утилиты:

- barrier - синхронизация процессов на группе машин (требует установки barrierd на каждой из нод);
- barrierd - daemon для barrier;
- jsd - daemon для запуска команд на удаленной машине (требует установки на каждую из нод);
- jsh - запуск команд на удаленной машине;

Примеры использования clusterit:

- Проверка на наличие уязвимостей для FreeBSD:

```
dsh -e -g freebsd "sudo /usr/local/sbin/portaudit -Fda"
```

- Проверка уязвимостей для Linux Gentoo:

```
dsh -e -g gentoo,vds "glsa-check -ln afected"
```

Еще одним критичным вопросом является резервное копирование. Для бекапов данных сервера у каждого администратора находится свое решение. Для облегчения жизни и спокойного сна автор хотел содержать центральное хранилище для файлов конфигураций со всех серверов и получать уведомления в случае, если файлы изменились без его участия, если системами управляет более одного инженера. Для этого была написана утилита backup2svn (http://backup2svn.sf.net/). Как backend в ней используется система хранения версий Subversion, а как транспорт для передачи файлов от удаленного сервера на сервер бекапов используется OpenSSH. Данное решение является несложным в использовании и настройке. Его дополнительный плюс - отсутствие необходимости устанавливать дополнительные утилиты со стороны клиента.

Примеры использования backup2svn:

```
Начинаем backup:
       $SSH_CMD bmw.kiev.ua \
       sudo gtar -czf - \
         /etc \
         /usr/local/etc | /opt/sbin/backup2svn bmw.kiev.ua
Изменился файл  /etc/ccd.conf:
Index: etc/ccd.conf
===================================================================
--- etc/ccd.conf        (revision 102)
+++ etc/ccd.conf        (working copy)
@@ -4,7 +4,7 @@
 +ccd0  16      none    /dev/sd2e /dev/sd3e
 -ccd0   16      none    /dev/wd0a
```

Платформа разработчика OpenMoko

Александр Боровский - ООО Скэнд - Минск, Беларусь

В работе выполнен обзор проекта OpenMoko, рассмотрены особенности интерфейса, связанные со спецификой устройств ввода-вывода. Рассмотрены способы разработки и отладки программ для платформы OpenMoko, в том числе на настольном компьютере.

Мобильные устройства характеризуются большим разнообразием устройств ввода (ограниченный набор кнопок, сенсорный экран, внешняя клавиатура и т.д.), ограниченным пространством на экране - на PC наоборот используется стандартный набор из клавиатуры, мыши и монитора - а также широким разнообразием архитектур процессора.

История

На данный момент существует несколько распространённых мобильных платформ, и конечно каждая из них имеет свои недостатки:

- Symbian - очень распространённая на смартфонах операционная система. Характеризуется высокой кастомизируемостью, имеет неплохую безопасность (однако системы защиты могут использоваться для сокрытия вредоносного кода), высокую эффективность использования ресурсов. Требует лицензирования.
- WinCE - распространённая система на смартфонах и еще более - на наладонниках. Хотя ядро очень эффективно, сама система отличается неторопливостью. Требует лицензирования.
- Palm - отличается высокой эффективностью и очень быстрой работой (однозадачная на пользовательском уровне и многозадачная на уровне ядра). Разработка PalmOS уже не ведётся.
- Linux на мобильных устройствах появился впервые на Zaurus SL-5000D (ноябрь 2001) и в виде Familiar linux (проект по разработке дистрибутива для наладонников Compaq iPAQ (февраль 2002). Затем наработки этих проектов объединились в проект OpenEmbedded (в феврале 2003).

OpenMoko

OpenMoko был представлен 7 ноября 2006 года. Система построена на базе OpenEmbedded (ядро 2.6, busybox, udev, blueZ, dbus, X Server, GTK+2, matchbox).

Поверх базовых сервисов написан набор библиотек, предоставляющих высокоуровневые функции для стандартных операций, таких как отправка SMS и звонок по заданному номеру (libmokocore); высокоуровневые сетевые функции, например отправка файла через bluetooth (libmokonet), работа с персональной информацией, контактами, календарями (libmokopim) и дополнительные элементы управления (libmokoui).

Интерфейс OpenMoko

Т.к. Neo1973 имеет только touchscreen и 2 кнопки, то навигация в интерфейсе производится с помощью стилуса или пальцев. Это отражено и в интерфейсе: есть приложения, управляемые стилусом (stylos-based), а есть - управляемые пальцами (finger-based, с увеличенными размерами элементов). Для каждого типа приложений характерен свой макет (layout) интерфейса и, частично, набор элементов управления.

Макет интерфейса для finger-based приложений может варьироваться, например в

calculator и dialer это панель с кнопками и небольшим экраном сверху, а для main menu это куча иконок с прокруткой с помощью специального колеса (отдельный элемент управления).

Макет интерфейса для stylos-based приложений более-менее одинаков: сверху меню приложения/фильтрации, под ним список элементов (navigation area), затем toolbar/searchbar; снизу расположена details area, которая может развёртываться на весь экран (кнопкой на scrollbar).

Neo1973

Neo1973 - это мобильный телефон для обкатки OpenMoko. Он имеет 200Mhz процессор (ARM920T), 128Mb RAM, 64Mb flash, GSM/GPRS, AGPS, высококонтрастный LCD дисплей (480x640), Unpowered USB host.

Все устройства доступны для программ: GSM и AGPS как обычные serial устройства, микрофон, стереовыход и динамик через ALSA, USB через стандартные драйвера.

Разработка на OpenMoko

Сейчас для разработки используется окружение OpenEmbedded. В качестве UI toolkit используется GTK+2 с дополнительными элементами управления.

Есть несколько способов отладки программ для OpenMoko: использовать Xoo (XNest), использовать эмуляцию в qemu или использовать debug board.

Xoo обладает наибольшей скоростью, однако почти никакие аппаратные особенности не эмулируются (выполняется приложение в chroot с ядром системы). QEmu позволяет тестировать приложения в более-менее приближенных к реальности условиях (в OpenMoko SDK идёт qemu, эмулирующий почти все устройства). Debug board подключается к реальному телефону (Neo1973), и идеально подходит для разработки драйверов.

Для упрощения разработки и сборки образов был разработан MokoMakefile, который позволяет:

1. Собирать образы дисков (под разные платформы);
2. Запускать эти образы в qemu;
3. Обновлять среду разработки (из которой собираются образы).

Литература

1. PDA, история http://en.wikipedia.org/
2. Информация о OpenMoko http://wiki.openmoko.com/

Maemo - платформа разработки программ для ОС Internet Tablet

Владислав Васильев УП "Витебскоблгаз" - Витебск, Беларусь
Павел Фиалко - НРУПТН "Дружба" - Новополоцк, Беларусь

В предлагаемом докладе описывается платформа Maemo - платформа разработки с открытыми исходными текстами для интернет-планшетов фирмы Nokia. Освещены следующие вопросы: особенности платформы, история создания и роль компании Nokia в ней; устройства, работающие на платформе Maemo; рассказано о сообществе и людях принимающих участие в продвижении платформы. Также рассмотрены среда и средства разработки платформы, механизмы портирования существующих программ, приведен обзор приложений уже работающих на данной платформе.

что такое Maemo и роль в ней Nokia

Maemo представляет собой платформу разработки с открытыми исходными текстами для интернет-планшетов Nokia и других, основанных на linux устройствах.

Платформа Maemo включают в себя инструменты необходимые для создания и портирования приложений для ОС Internet Tablet. Операционная система Internet Tablet представляет собой модифицированный вариант дистрибутива Debian GNU/Linux. Разработчику, знакомому с технологиями GTK+/GNOME и дистрибутивом Debian, не составит труда разобраться в Maemo.

Компания Nokia всячески способствует развитию платформы и направляет ее развитие. Она первая разработала устройства, использующие эту платформу.На данный момент это интернет-планшеты Nokia 770 и Nokia N800. Корпорация всячески стимулирует подключение новых разработчиков, дизайнеров и активных пользователей к работе над платформой.

какие устройства работают на этой платформе

Как было сказано выше, к настоящему моменту компания Nokia выпустила два устройства работающие на платформе Maemo - это Nokia 770 Nokia и N800.

Из основных характеристик устройств можно назвать использование ARM процессоров и большой 800x480 пикселей размер экрана. Время работы до 3 часов в рабочем режиме и до 7 часов в режиме ожидания. Время работы обоих устройств сильно зависит от количества программ запущенных программ. Надо акцентировать внимание на то, что эти устройства не является smartphone и не имеют gsm-модулей. Это принципиальная позиция Nokia. Возможно она связана с боязнью создания конкуренции своим же продуктам на базе Symbian OS.

Внешний вид интернет-планшетов приведен на рисунках:

[Рисунок 1. Внешний вид интернет-планшета Nokia 770]

[Рисунок 2. Внешний вид интернет-планшета Nokia N800]

сообщество Maemo

Сообщество Maemo очень разнообразно в географическом плане. Есть разработчики из Финляндии, России, Западной Европы, США и из Латинской Америки и по ориентировочным оценкам количество активных разработчиков составляет несколько сотен человек.

На официальном сайте проекта [1] содержится документация, учебные материалы, репозитарий программ для тестирования. Также сайт maemo.org предоставляет услуги для сообщества Maemo, куда входят:

- каталог программ;
- хостинг проектов сторонних разработчиков под платформу Maemo [2];
- электронная энциклопедия wiki;
- новостной портал Maemo.

Также есть irc канал и ряд форумов.

среда и средства разработки платформы Maemo

На данный момент существует две основные ветки SDK [3] для устройств, работающих на платформе Maemo. Первая ветвь - Maemo 2.2 'gregale' - предназначена для разработки программ для интернет-планшета Nokia 770, работающих под управление IT OS, редакции 2006 года. Вторая ветвь - Maemo 3.1 'bora' - предназначена для разработки программ для интернет-планшета Nokia N800, работающих под управление IT OS, редакции 2007 года, версии 3.2007.10-7.

Обе ветви SDK представляют собой так называемые "rootstrap" - образы корневой файловой системы устройств. Причем для Nokia 770 образ разбит на два архива, первый - пакеты для разработки корневой файловой система (используется только на Nokia 770), второй - набор пакетов для SDK. Для Nokia N800 существуют rootstrap для двух архитектур - i386 и armel.

Обе ветви дополнительно требуют для работы кросс-платформенный набор инструментов Scratchbox R4 или более поздней. Официальный сайт Scratchbox - scratchbox.org.

В качестве средства для построения графического интерфейса пользователя платформа Maemo использует библиотеку GTK+. Основным языком для создания приложений на платформе Maemo является Си. Также возможна разработка программ и на языке Питон. Но разработчикам на Питоне следуют иметь в виду некоторые ограничения, накладываемые платформой Maemo (для подробной информации необходимо обратиться к документации [4]).

Из инструментов отладки приложений на платформе Maemo присутствуют следующие:

- gdb - отладчик;
- strace - трассировщик системных вызовов;
- valgrind - отладчик и профилер, работает только для архитектуры x86 и только внутри Scratchbox.

Дополнительная информация по применению инструментов отладки содержится в "maemo debugging guide" [5].

возможность портирования

При портирование приложений на платформу Maemo существуют некоторые ограничения.

В основном все ограничения по портированию приложений касаются интерфейса пользователя и используемых библиотек. Этапы процесса портирования сильно зависят от

портируемого приложения, от тех библиотек и функций которые используются в нем. Однако общий алгоритм портирования можно описать так:

- удалить или заменить все не поддерживаемые платформой библиотеки;
- исправить или переписать все функции использующие удаленные библиотеки;
- модифицировать интерфейс пользователя с учетом требований платформы, например,
    - разрешение экрана не должно превышать 800x480 точек;
    - исправить структуру дерева файлов портируемой программы;
    - внести изменения или создать файлы отвечающие, за формирование пакета Debian, руководство по созданию пакетов можно найти по адресу http://packages.debian.org/

Пример портирования приложения и инструкции по портированию приведен в [6].

краткий обзор программ, работающих под платформой Maemo

Все приложения уже работающих под рассматриваемой платформой можно квалифицировать по следующим критериям:;

- Идущие в стандартной поставке;
- Приложения сторонних разработчиков;
- Портированные приложения;
- Приложения специально написанные под Maemo и другие платформы работающие с наладонными компьютерами.

Ссылки

1. Официальный сайт проекта Maemo // http://maemo.org
2. Хостинг проектов под платформу Maemo // http://garage.maemo.org
3. Maemo SDK // http://maemo.org/development/sdks/
4. Особенности Python для Maemo // http://maemo.org/development/documentation/how-tos/3-x/python_maemo_3.x_howto.html
5. Maemo debugging guide // http://maemo.org/development/documentation/how-tos/3-x/maemo_debugging_guide.html
6. Инструкции по портированию приложений // http://maemo.org/development/documentation/how-tos/3-x/howto_porting_to_maemo_bora.html

Организация процесса разработки программных продуктов с использованием свободного ПО

Борис Турчик - ООО "Интеграл" (IPonWeb) - Москва, Россия

В докладе автор делится собственным опытом разработки программных продуктов и организации рабочего процесса при помощи свободного ПО. Кратко рассмотрены основные принципы разработки, организации командной работы и распределения задач, а так же контроля над версиями и цельностью исходного кода. Доклад может быть интересен как планирующим разрабатывать ПО, так и уже занимающимся разработкой.

ОСНОВЫ ОРГАНИЗАЦИИ РАЗРАБОТКИ ПО

На этапе организации определяются основные цели, задачи, а так же основной инструментарий, который будет использоваться в процессе разработки. Так же обозначается определённая структура проекта, распределение областей разработки и общие соглашения по написанию кода проекта и технической документации.

Кроме того, необходимо учесть что в силу определённых причин некоторые изначально установленные рамки могут быть изменены в дальнейшем. К примеру, инструментарий разработки может быть (и, скорей всего, будет) дополнен или изменён, равно как может быть изменена и структура проекта.

ИНСТРУМЕНТАРИЙ РАЗРАБОТЧИКА

Существует довольно обширный выбор различных инструментов для выполнения поставленных задач. Наш проект изначально был ориентирован на использование свободного ПО при разработке веб-сервисов. Необходимо было подобрать подходящие продукты для следующих задач:

- платформа разработки;
- система контроля версий;
- система управления и распределения заданий;
- система автоматического тестирования.
- СУБД

В силу различных причин были выбраны следующие программные продукты:

- PHP 5.x как основная платформа для разработки;
- Subversion в качестве системы контроля версий;
- RT в качестве управления заданиями;
- Система автоматического тестирования была реализована самостоятельно на основе уже существовавшего кода функциональных тестов для смежного проекта. Для реализации использовался Perl;
- MySQL 5.x как СУБД.

В дальнейшем список используемых средств значительно расширился в силу необходимости поддержки различных клиентских платформ и расширения функциональности.

Выбор операционной системы и редактора остался за разработчиком, при условии что оба продукта будут свободными. В результате чего на проекте используются Debian Sarge и Gentoo в качестве операционных систем (а так же CentOS на рабочих серверах), и Emacs/KDevelop/Eclipse в качестве редакторов.

СОГЛАШЕНИЯ ПО РАЗРАБОТКЕ

При разработке любого программного продукта устанавливаются определённые соглашения по написанию, тестированию и отладке кода. Код, который не соответствует заданным (или хотя бы базовым) стандартам, довольно сложен для чтения и понимания, что неприемлемо при командной работе, когда несколько человек поддерживают один код или некоторые общие участки. Кроме того, автоматическое тестирование кода играет крайне важную роль, поскольку вероятность присутствия багов и недочётов в коде весьма высока. Так же довольно важную роль играют комментарии и документирование кода - комментарии для описания наиболее магических мест в коде и документирование для автогенерации технической документации.

При написании кода использовались слегка модифицированные соглашения PEAR, т.к. основной код писался на PHP5, так же аналогичные правила применялись к написанию тестов. В общие правила входило так же требование написания тестов к любому изменяемому/дополняемому коду, т.е. никакая логика не должна была попасть в систему без тестов. Недостатком в данном случае являлось то, что код и тесты реализовывались одним и тем же разработчиком (за редкими исключениями). Однако этот недостаток восполнялся тем, что весь код отдавался на проверку старшим разработчикам, которые следили за тем, чтобы логика была надлежащим образом покрыта тестами. В случае если код был написан одним из старших разработчиков, он передавался на проверку другому. Следует также заметить, что в идеале тесты должны создаваться до того, как будет написана логика. К сожалению, такой стратегии сложно придерживаться, когда логика и тесты к ней создаются одним разработчиком.

Кроме того, в обязанности разработчика входило базовое документирование логики, т.е написание комментариев в особо сложных местах и краткое описание методов, их параметров и свойств классов. Более подробное описание выполнялось отдельным разработчиком, ответственным за документацию и примеры использования.

СТРУКТУРА ПРОЕКТА

Для распределения обязанностей и контроля над качеством и правильностью кода в команде должны присутствовать старшие или более опытные разработчики, в обязанности которых входит обсуждение функционала, постановка задачи и реализация и/или проверка написанного кода. Код, попадающий в работающую систему без дополнительной проверки, рискует содержать в себе недочёты, даже при условии его автоматического тестирования. В идеале написание тестов и собственно разработка кода должна выполнятся различными людьми или даже различными командами на основе одной спецификации - таким образом можно исключить допуск одинаковых ошибок в коде и в тестах.

Исходя из этого, структура выглядит следующим образом:

- тимлидер - осуществляет распределение частей проекта между старшими разработчиками, просмотр кода, отслеживает сроки выполнения заданий а так же создание новых при необходимости;
- старшие разработчики во главе с тимлидером - создают спецификации, основываясь на предложениях самих разработчиков или пожеланиях заказчика. Старшие разработчики распределяют задания среди младших или выполняют сами, после чего проверяют результаты или передают другим (или тимлидеру) для проверки. Так же задание может быть передано другой команде (скажем, системным администраторам) для выполнения промежуточных действий. Задачи старших разработчиков так же могут различаться в зависимости от роли в команде - релиз-менеджер, поддержка пользователей и т.д.);
- младшие разработчики - осуществляют непосредственно разработку кода заданий, полученных от старших разработчиков.

ЭТАПЫ РАЗРАБОТКИ

Разработка программного продукта должна быть разделена на определённые этапы. Это необходимо как для определения сроков выполнения той или иной части разработки, так и для удобства разработчиков. Обычно задание проходит этапы спецификации, разработки и проверки, а так же непосредственно этап попадания в рабочую систему. Но довольно часто случаются ситуации, при которых данная последовательность невозможна. Точнее говоря, в нее включается несколько дополнительных этапов. Например, такое может происходить при кардинальном изменении различными разработчиками одного участка кода или если изменения затрагивают слишком большую область кода.

Поэтому необходимо не только разделить процесс разработки на этапы, но и предусмотреть возможность внесения корректив и дополнений, а также вероятные случаи возникновения таковых возможностей.

Базовые этапы разработки выглядят следующим образом:

- Создаётся задание в соответствующей очереди RT;
- задание отдаётся на выполнение разработчику, присваивается в RT;
- разработчик, используя команды svn, копирует существующий код в отдельную ветку с номером задания;
- разработчик выполняет задание в открытой ветке. таким образом, все его изменения не будут вилять на разработку в других ветвях;
- если в корневом репозитории произошли изменения, затрагивающие разработку, разработчик может синхронизировать свою ветвь с корневой;
- разработчик передаёт выполненное задание старшему, переназначая его в RT;
- старший разработчик соединяет код ветви с корневым кодом, просматривает разницу кода, тестов, а так же запускает тесты;
- при наличии конфликтов или недочётов в коде задание отдаётся обратно разработчику;
- если старший разработчик подтверждает изменения, он помещает их в корневой svn и, при необходимости, в предыдущие версии;
- в случае если код затрагивает предыдущие версии, задание передаётся релиз-менеджеру для помещения в следующий релиз.

Разработка локализованного дистрибутива Linux с широкой функциональностью для научных и образовательных учреждений с максимальной простотой внедрения

Святослав Грыцько - Львов, Украина

Рассматривается метод создания специализированного сборника свободного программного обеспечения, способного работать как с LiveCD, так и с локального диска, а также загружать бездисковые узлы через сеть.

Работая с дистрибутивом Gentoo несколько последних лет, я столкнулся с, можно сказать, "некоторым неудобством" его установки в новом месте и демонстрации возможностей в кругу, где обычного знакомого программного окружения нет. Мне хотелось иметь инструмент, полностью клонирующий привычную мне рабочую систему с локального диска, при этом сжимая её и создавая ISO-образ, которым можно было бы пользоваться как LiveCD/DVD или USB-FLASH и потом при желании иметь возможность воссоздать с полученного LiveCD клон системы обратно на локальный диск. Также такой LiveCD/DVD можно было бы использовать как рабочую резервную копию системы.

Стоит отметить, что система на диске и клон на LiveCD создаются идентичными за исключением параметров загрузки ядра, которыми и указывается факт загрузки с LiveCD. Конечно система на локальном диске должна быть специально собранной - в ядро должна быть включена необходимая функциональность, а также надо надлежащим образом собрать первичный загрузочный диск initrd и пару специальных приложений. Это к тому же очень способствует и умению системы загружается через сеть на бездисковых узлах.

Была сделана демонстрационная версия CD, в которую вошли лучшие программы для программирования, системного администрирования, физики, математики, химии, бухгалтерии и офисной или домашней работы (данный сборник будет весьма полезен для научных работников, преподавателей, учителей, аспирантов, студентов и учеников; пока в сборник входят программы, ориентированные на компьютерные и естественные специальности, но в будущем можно добавить и гуманитарные направления).

Многие пользователи изъявляли желание установить систему локально. Для этого был разработан скрипт обратного клонирования LiveCD на локальный диск. Практика показала, что подавляющие большинство учителей информатики (чтобы не сказать все) не могут правильно переразбить, отформатировать и примонтировать диски для установки, а их попытки сделать систему с двойной загрузкой без сторонней помощи успехом не заканчиваются (отсутствуют минимальные навыки администрирования *nix), приводя к потере данных на диске. Скрипт был доработан таким образом, чтобы по умолчанию самостоятельно устанавливался на первый физический диск в системе, просто снося все данные. Также в его функции входит установка системы на узел загруженный по сети. После установки получается нормальная полнофункциональная система, идентичная первичной. Таким образом удалось сделать сборник максимально простым в использовании: установка на локальный диск "одним кликом мышки", возможность сразу приступать к работе без дополнительных настроек (система настраивается автоматически).

Таким образом появился DYSTRYK (ДИСТРИК) - дистрибутив свободного программного обеспечения, разработанный специально для использования в образовательных и научных учреждениях с поддержкой российской, украинской, белорусской и английской локализации.

Хотя для DYSTRYK в качестве базы использован Gentoo, но все скрипты можно применить и для любого другого дистрибутива Линукса, предварительно пересобрав ядро с initrd и пару-другую пакетов.

Изначально идея скрипта, который бы создавал ISO-образ LiveCD с установленной локально системы, взята с http://www.xnfo.org/scripts/build и немного переработана, чтобы снимался образ прямо с рабочей системы.

На сегодня перед "DYSTRYK" стоят следующие цели (отчасти решены):

1. Создание максимально удобного в использовании научно-образовательного локализованного "дистрибутиву" для рядового пользователя.

2. Возможность загрузки как с LiveCD так и с локального диска.

3. Обеспечение безопасности, приватности и анонимности пользователей.

4. Возможность загрузки бездисковых станций, узлов по сети (загружается один компьютер в классе; остальные компьютеры грузятся с него) и установка по сети на локальный диск "одним кликом мышки".

5. Многотерминальность. Возможность одновременной работы на одном системном блоке нескольких пользователей (многоголовые системы)

6. Кластерность. Быстрое автоматизированное развертывания кластеров разного функционального назначения:

6.1 Параллелизм - параллельное выполнения ресурсоёмких математических и других вычислений на многих процессорах.

6.2 Кластеризация дисков для обеспечения надежного и удобного хранения данных.

6.3 Автоматическая балансировка нагрузки на узлы кластера.

7. Разработка/внедрение программ для автоматизированного администрирования разветвленной инфраструктуры кластеров.

ПОСЛЕСЛОВИЕ

Использование свободного программного обеспечения - это экономия миллиардов рублей государственного бюджета, ускорение развития информационных технологий и смежных отраслей, обеспечение гарантий информационной безопасности и информационного суверенитета страны, переход государства в высший информационный уровень развития общества.

ИСТОЧНИКИ

"DYSTRYK", созданный на основе метадистрибутива Gentoo с помощью программ (инструкций для автоматизированного сбора) DYSTRYK
https://sourceforge.net/project/showfiles.php?group_id=116780&package_id=211038

http://moikrug.ru/m

Open Source Software in Schools

Shogbeni Abiola Surajudeen - Lagos, Nigeria

There are general reasons why all computer users should insist on free software. It gives users the freedom to control their own computers -- with proprietary software, the computer does what the software owner wants it to do, not what the software user wants it to do. Free software also gives users the freedom to cooperate with each other, to lead an upright life. These reasons apply to schools as they do to everyone.

But there are special reasons that apply to schools. They are the subject of this article.

First, free software can save the schools money. Even in the richest countries, schools are short of money. Free software gives schools, like other users, the freedom to copy and redistribute the software, so the school system can make copies for all the computers they have. In poor countries, this can help close the digital divide.

This obvious reason, while important, is rather shallow. And proprietary software developers can eliminate this disadvantage by donating copies to the schools. (Watch out! -- a school that accepts this offer may have to pay for future upgrades). So let's look at the deeper reasons.

School should teach students ways of life that will benefit society as a whole. They should promote the use of free software just as they promote recycling. If schools teach students free software, then the students will use free software after they graduate. This will help society as a whole escape from being dominated (and gouged) by megacorporations. Those corporations offer free samples to schools for the same reason tobacco companies distribute free cigarettes: to get children addicted. They will not give discounts to these students once they grow up and graduate.

Free software permits students to learn how software works. When students reach their teens, some of them want to learn everything there is to know about their computer system and its software. That is the age when people who will be good programmers should learn it. To learn to write software well, students need to read a lot of code and write a lot of code. They need to read and understand real programs that people really use. They will be intensely curious to read the source code of the programs that they use every day.

Proprietary software rejects their thirst for knowledge: it says, “The knowledge you want is a secret--learning is forbidden!” Free software encourages everyone to learn. The free software community rejects the “priesthood of technology”, which keeps the general public in ignorance of how technology works; we encourage students of any age and situation to read the source code and learn as much as they want to know. Schools that use free software will enable gifted programming students to advance.

The next reason for using free software in schools is on an even deeper level. We expect schools to teach students basic facts, and useful skills, but that is not their whole job. The most fundamental mission of schools is to teach people to be good citizens and good neighbors--to cooperate with others who need their help. In the area of computers, this means teaching them to share software. Elementary schools, above all, should tell their pupils, “If you bring software to school, you must share it with the other children”. Of course, the school must practice what it preaches: all the software installed by the school should be available for students to copy, take home, and redistribute further.

Teaching the students to use free software, and to participate in the free software community, is a hands-on civics lesson. It also teaches students the role model of public service rather than that of tycoons. All levels of school should use free software.

Оценка Open Source проекта по CMMI

Василий В. Савин - Вильнюс, Литва

В последнее время Открытое ПО становится всё популярнее. Правительства многих стран, крупные корпорации стали рассматривать Linux, OpenOffice.org и другое свободное ПО как достойную альтернативу коммерческим программам. Однако сам процесс создания свободного ПО изучен достаточно плохо. Пожалуй, единственная широко известная работа по данной тематике - «Собор и Базар» Эрика Рэймонда[1]. К сожалению, данная работа была написана 10 лет назад и на сегодняшний день несколько устарела. А другие авторы в основном изучали социальные и экономические аспекты OS проектов[2]. И только в последние несколько лет появилось несколько работ, где открытое ПО рассматривается с точки зрения инженерии ПО. В своём докладе я бы хотел представить выводы, полученные в процессе написания дипломной работы. Целью моей работы было изучить, насколько возможна оценка Open Source проектов по модели CMMI.

Что такое CMMI[2]?

CMMI (Capability Maturity Model) был создан институтом инженерии ПО по заказу минобороны США. Он был призван решить задачу оценки зрелости процессов и повысить эффективность данных процессов. На данный момент CMMI является de facto стандартом оптимизации процессов по созданию ПО. Внедрение CMMI позволило организациям добиться значительного улучшения качества, уменьшить количество дефектов и сократить время разработки.

Существует несколько разных моделей – для управления субподрядчиками, закупками (Acquisition) и наиболее интересного с точки зрения разработчика ПО - модель процесса разработки систем (CMMI for Development). Данная модель позволяет описать любой процесс разработки, в том числе и ПО. Для этого весь жизненный цикл разбивается на 4 категории и 22 области процесса

Что такое Agile[3] методики?

Agile стал ответом программистского сообщества на попытку навязать традиционный подход к созданию ПО. По мнению сторонников Agile, попытки устранить из проектов неопределённость заранее обречены на неудачу. Так как даже в идеальном случае, если клиент получит ту систему, которую он заказывал полгода назад, то совершенно необязательно она удовлетворит его потребности сегодня и будет пригодна для расширения в нужном для него направлении.

Agile предлагает не устранять неопределённость, а её контролировать путём более коротких итераций, фиксацией сроков, а не функциональности и более тесного контакта с клиентом.

Выводы и результаты. Изучая модели процессов создания свободного ПО в различных проектах, я пришёл к выводу, что единого процесса, характерного для всех Open Source проектов не существует, так как проекты достаточно сильно отличаются друг от друга: размерами, задачами, целевой аудиторией.

Поэтому моей первой задачей стало создать классификацию процессов создания ПО а Open Source проектах. В результате были выделен ряд параметров, на основании которых возможно объединить проекты в эквивалентные классы:

- **Разработчики**: один или команда
- **Местоположение команды**: централизованно\локально или распределенно\глобально

- **Целевая аудитория**: конкретный заказчик или массовый потребитель
- **Коммерциализация**: коммерческий или некоммерческий проект
- **Степень сотрудничества**: близкое\плотное или слабое

Охарактеризовав проект свободного ПО по этим параметрам можно составить достаточно точный портрет данного проекта.

Следующей моей задачей стало оценить проект свободного ПО по CMMI. Однако прямой подход - дословно интерпретировать требования модели, бесперспективен, так как ряд задач в проектах свободного ПО достигается иными способами. Кроме того ряд описываемых практик малополезен для проектов свободного ПО. Поэтому были изучены способы оценки Agile методик, к которым можно причислить и модель создания открытого ПО.

В качестве эксперимента я оценивал проект plone.org по следующим областям:

- Планирование проекта
- Управление проектом
- Управление конфигурацией
- Измерения и анализ
- Обеспечение качества процесса
- Определение требований пользователей

Ссылки:

1. http://www.catb.org/~esr/writings/cathedral-bazaar/
2. http://opensource.mit.edu/online_papers.php
3. http://www.sei.cmu.edu/cmmi/
4. http://en.wikipedia.org/wiki/Agile_software_development

Глобальное применение Open Source и Freeware систем для глобальных корпоративных информационных систем

Василий К. Савин - Вильнюс, Литва

Рассматривается опыт использования открытого и бесплатного ПО на всех уровнях (от серверов и глобальной сети до рабочих мест) для построения и эксплуатации корпоративных информационных систем.

Введение

Открытое ПО в силу своей специфики позволяет максимально оптимизировать систему под имеющуюся аппаратуру и желаемые характеристики, т.е. получить наилучшее соотношение цена/качество, что дает возможность, в свою очередь, либо снизить стоимость, либо получить лучшие характеристики системы. Бесплатное ПО, предоставляемое некоторыми компаниями, обычно не обладает всеми возможностями коммерческого ПО тех же компаний, а также не дает той гибкости, которая присуще открытому ПО. Тем не менее при наличии соответствующей квалификации разработчика и/или администратора системы, бесплатное ПО позволяет существенно расширить функциональные возможности системы без существенного увеличения стоимости.

Типовая схема корпоративной информационной системы выглядит следующим образом:

[global_is.jpg]

Современный уровень развития открытого ПО и предлагаемого различными коммерческими компаниями бесплатного ПО позволяет строить полноценные и эффективные корпоративные информационные системы, а также обеспечить службу сопровождения и поддержки соответствующим инструментарием для оперативного контроля и управления (администрирования) таких систем.

Далее рассмотрим соответствующие решения на каждом из уровней корпоративных информационные систем.

Серверные фермы

Современные принципы построения корпоративных северных ферм демонстрируют устойчивую тенденцию к конвергенции и виртуализации как самих серверов, так и устройств хранения данных. Многие производители серверов (IBM, Sun) имеют собственные средства виртуализации. Среди Open Source и Freeware решений, которые мы здесь рассматриваем, также существует немало интересных проектов (OpenVZ, Qemu, VirtualBox, Xen). Поскольку перед нами стояла задача обеспечить хостинг с приемлемой производительностью не только для Linux-серверов, но и для Windows-серверов, с учетом ограничений аппаратуры сервера, не поддерживающей виртуализацию на аппаратном уровне, выбор резко сужается, и практически остается единственный реальный кандидат – Vmware Server[2]. Это достаточно широко распространенный преемник не менее известного коммерческого продукта той же компании Vmware GSX Server, который чуть более года назад был предоставлен компанией Vmware для бесплатного пользования.

Рекомендуется следующая схема подключения реальных и виртуальных серверов:

[vm-hosts.jpg]

Такое решение обеспечивает работу виртуальных серверов при единичном отказе любого из реальных компонентов системы.

Подключение виртуальных серверов к сети

Чаще всего корпоративные сервера не находятся в одном сегменте сети, а логически разделены для обеспечения безопасности, поддержания внутренней инфраструктуры и т.д. Схема подключения серверов обычно выглядит так:

[vlans.jpg]

Здесь возможны проблемы, если количество виртуальных сегментов сети превышает количество реальных портов для подключения к сети, поскольку Vmware Server не поддерживает виртуальных сетевых коммутаторов, - эта возможность реализована только в коммерческой версии Vmware ESX Server. И тут уже проявляется мощь открытого ПО, - мы просто включаем поддержку протокола 802.1q в Linux-ядре[1] и настраиваем систему для работы с VLAN'ами[3]!

Маршрутизаторы и firewall'ы

Общеизвестно, что эти функции давно и весьма успешно выполняют подсистемы на базе открытого ПО. Наиболее популярны для этих целей iptables[4] и iproute2[6]. Менее известен ebtables[5], который также порой полезен.

Удаленная связь

При создании системы связи с филиалами, возникают специфические проблемы, которые приходится дополнительно решать:

- приоритетизацию сетевого трафика (QoS[7]);
- туннелирование и шифрование канала связи (Ipsec[8]);
- надежность канала связи (dial-on-demand) путем использования альтернативных каналов (ISDN, EDGE, 3G etc) прозрачно для пользователей;

Схема глобальной сети обычно выглядит так:

[wan.jpg]

Здесь также уже давно и успешно применяются решения на базе открытого ПО.

Серверы филиалов

Схема типового филиала:

[branch.jpg]

Задачи серверов филиала обеспечить:

- доступ пользователей филиалов к локальной информации (samba[9]);
- синхронизацию (rsync[10]) локальной информации с централизованным хранилищем;
- локальную почту;
- локальный иерархический proxy[11] с фильтрацией[12];
- защиту от вирусов;

Это также общеизвестные, «классические» применения открытого ПО, за исключением сервиса синхронизации данных, который распространен в меньшей степени.

Рабочие места

В реализованных системах применялись следующие виды рабочих мест (в зависимости от требований заказчика):

[clients.jpg]

Обычные рабочие места:

- на базе Windows (Win98, Win2K, WinXP);
- на базе Linux (KDE/Gnome/IceWM/XFce);

Бездисковые рабочие места с загрузкой от CD/PXE:

- на базе Linux (IceWM/XFce);
- на базе Linux (клиент M$ терминал-сервера);
- Интернет-киоск на базе Linux (Firefox с ограниченным списком допустимых сайтов);

Бездисковые рабочие места с flash-memory:

- на базе Windows (WinCE, WinXP-E);
- на базе Linux (XFce);

Мобильный удаленный клиент (notebook) с OpenVPN доступом.

Подсистема управления, контроля и оповещения

Полноценная промышленная эксплуатация любой системы невозможна без действенной подсистемы управления, контроля и оповещения. Эта подсистема должна включать в себя не только средства отображения контролируемых параметров системы, но и средства их накопления и генерации отчетов. В нашей практике для оперативного управления и контроля различных подсистем в основном используются системы на базе веб-технологий, как наиболее удобные и доступные администратору из любой точки сети без специального клиентского ПО.

- Управление:
  - Для виртуальных серверов это штатные средства Vmware (vmware-mui);
  - для большинства других систем – webmin[13];
  - конечно же shell для *NIX в разных видах (по сети, через мобильный телефон, через webmin etc);
  - специализированные средства;
- Контроль:
  - жизнеспособности всех подсистем - Nagios[14];
  - состояния каналов передачи данных – всем хорошо известный MRTG[15];
- Оповещение:
  - Аудио-визуальные средства
  - Электронная почта
  - SMS-сообщения
- Накопление:
  - системные журналы
  - базы данных

В целом структура контроля имеет следующий вид:

[monitoring.jpg]

В настоящее время ведутся испытания еще нескольких средств контроля состояния систем, которые возможно заменят некоторые ныне существующие.

Заключение

Многолетняя эксплуатация систем различного назначения и различного уровня сложности, как исключительно на базе открытого ПО, так и в различных сочетаниях с бесплатным и коммерческим ПО, показывает реальность и жизнеспособность таких решений, и даже их техническое и финансовое превосходство. Однако, к сожалению, на корпоративном рынке эти решения сталкиваются со следующими проблемами (человеческий фактор?!):

- нежелание дистрибьюторов продавать эти решения из-за низких комиссионных;
- повышенные требования к квалификации разработчика и/или администратора системы;

- повышенная ответственность администратора системы.

Источники

1. The Linux Kernel Archives: http://www.kernel.org/
2. Vmware server: http://www.vmware.com/products/server/
3. 802.1Q VLAN implementation for Linux: http://www.candelatech.com/~greear/vlan.html
4. Linux kernel firewall, NAT and packet mangling tools: http://www.iptables.org/
5. Utility that enables basic Ethernet frame filtering on a Linux bridge, MAC NAT and brouting: http://ebtables.sourceforge.net/
6. Linux Advanced Routing & Traffic Control HOWTO: http://lartc.org/howto/
7. iproute2+tc: http://snafu.freedom.org/linux2.2/iproute-notes.html
8. The OpenSource IPsec-based VPN Solution for Linux: http://www.strongswan.org/
9. SAMBA is a suite of SMB and CIFS client/server programs for UNIX: http://www.samba.org/
10. File transfer program to keep remote files into sync: http://rsync.samba.org/
11. A full-featured web proxy cache: http://www.squid-cache.org/
12. Web content filtering via proxy: http://dansguardian.org/
13. Web-based system administration interface: http://www.webmin.com/
14. Host, service and network monitoring program: http://www.nagios.org/
15. A tool to monitor the traffic load on network links: http://oss.oetiker.ch/mrtg/

Организация и администрирование домашних ЛВС с доступом к провайдерам на базе ОС Линукс

Виктор Краев - ОИТ МИТСО - Минск, Беларусь

Рассмотрены особенности технической реализации домашней ЛВС в микрорайоне "Масюковщина". На ее примере описана организация бюджетного варианта ЛВС на неуправляемых свичах с центральным маршрутизатором/сегментатором на базе ОС Debian. Рассмотрена организация воздушных линий и бюджетный вариант их грозозащиты, организация связи по радиоканалу между сетями. Обосновывается использование открытой биллинговой системы Stargazer для управления доступом пользователей к сети и провайдерам. Рассмотрены средства мониторинга сети.

Организация ЛВС

Одна из причин появления домашних ЛВС - отсутствие полноценного интернета для домашнего пользования (соединение по dial-up в расчет не берется как издевательство над человеком по цене и качеству). Соединение по технологии xDSL для личного пользования до сих пор остается весьма дорогостоящим, из-за чего сам собой напрашивается выход – объединяться и совместно оплачивать то, что не по карману одному человеку. Тут на помощь приходят вторичные провайдеры, предоставляющие доступ по ADSL уже организованным сетям.

Географическое положение разработанной сети таково, что с 1-ой стороны – воинская часть, со 2-ой железная дорога и поле, с 3-ей (в сторону центра города) – поле с высоковольтной линией передач, 4-ой деревня, за ней поле и кольцевая. В 2004 году с помощью компании "Solo" появилась возможность подвести Интернет по выделенной линии, но для этого требовалось определенное количество пользователей. Тогда и появилась описываемая сеть.

Микрорайон в основном состоит их пятиэтажных хрущевок. На один 4-6 подъездный дом приходится по 10-15 пользователей. Есть даже 2-х этажки. Из-за специфики микрорайона использовать подземные коммуникации не представлялось возможным, поэтому сеть растянулась на 18 домов и 17 воздушных линий при 200 пользователях. Выделенная линия оказалась длинной 4 км, максимальная скорость - 264 кбит. Когда с ростом сети скорости стало не хватать, нашли пару старых таксофонных линий (сами таксофоны давно снесли), по которым и подключили 3 провайдера.

Бюджетный вариант на неуправляемых свичах с центральным маршрутизатором/сегментатором на базе ОС Debian

При выборе сетевого оборудования вариант использовать умный 24-портовый коммутатор на подъезд/дом (как в сетях, где длинные 9ти этажные дома) был отброшен. При таком огромном количестве воздушных линий вероятность выхода из строя сетевых коммутаторов из-за статики во время грозы, штормовых ветров и пр наводок очень велика. В одну из первых серьезных гроз вынесло около 8 коммутаторов (цепная реакция).

Естественно, что менять дорогостоящие управляемые коммутаторы - задача, несовместимая с бюджетом такой сети. Поэтому решили остановиться на недорогих, но качественных неуправляемых свичах Planet FSB-803 (8 портов) и FSB-1603 (16 портов). Задачи по управлению трафиком, контролю MAC-адресов, IP-адресов, фильтр бродкаста и пр. мусора, и самое главное – контроль доступа к провайдеру и остальным сегментам сети были возложены на центральный компьютер, выполняющий роль маршрутизатора и делящий сеть на сегменты (в нашем случае – 4 сегмента). Использован вариант с 2 процессорами pIII-

800, 1 гигабайт ОЗУ, 4 сетевыми картами, стоимостью 170 у.е. В качестве ОС выбор пал на Linux Debian (основная мотивация – наличие бесплатного доступа к репозитарию и наличие друзей-гуру.

На нем работают:

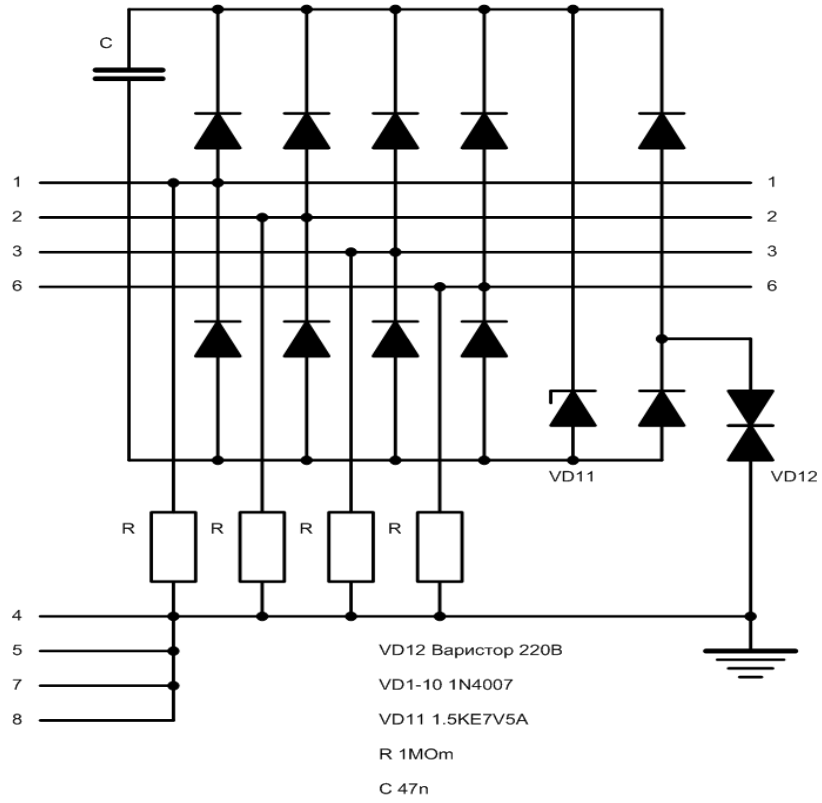
- DHCP-сервер (isc-dhcp3-server) - автоматическая рассылка сетевых параметров для windows-клиентов;
- WINS-сервер (Samba3) - сеть Windows;
- DNS (bind9) (лучше вынести на отдельный сервер) - был поднят DDNS (динамический для локальной сети) с форвардом на DNS провайдера.

Воздушные линии. Грозозащита.

На начальном этапе проектирования сети стоит сразу обозначить участки, где магистральный провод будет идти по крыше, фасаду и между домами по воздуху. Имеет смысл использовать на этих участках специальный кабель с экраном. В нашем случае использовался 4-х жильный военно-полевой кабель П-296. Особенность этого кабеля в том, что он имеет толстые жилы и отличный экран (можно даже вешать без троса на небольших расстояниях).

Необходима защита обоих концов (в некоторых случаях лучше ставить одну; о них см. ниже) воздушных линий грозозащитами. Грозозащиты можно купить настоящие (т.е. заводские) но их стоимость от 40 у.е. не позволяет говорить о бюджетном варианте =)

Можно купить грозозащиты кустарного производства – 6-10 у.е. или сделать их самостоятельно по себестоимости порядка 3 у.е:



Обязательно необходимо предусмотреть заземление грозозащит, экрана воздушной линии и ящиков, в которых находится сетевое оборудование.

Желательно на особо дорогих девайсах поставить бесперебойные блоки питания.

Радиолинк

Необходимо было обеспечить связь между двумя сетями, находящимися на расстоянии 2,5 км. Для обеспечения прямой видимости антенны точек доступа пришлось устанавливать на крышах двух девятиэтажных зданий. Для того, чтобы избежать потерь в коаксиальном кабеле, точки доступа установили на минимальном расстоянии от антенн, так как расстояние от одной из точек доступа до коммутатора составляло более 30 метров.

В качестве точек доступа можно выбирать любые Wi-Fi-устройства, как внешние, так и внутренние с разъемом для антенны. Изначально мы остановили выбор на внешних точках доступа D-Link; сейчас используются Planet WAP4030A со скоростью 54 мбит. Следует сразу обратить внимание на цифры 11/22/54/108, в которых кроется подвох. Так как все Wi-Fi устройства полудуплексные, чтобы выявить реальную скорость, следует разделить указанное число на 2.

В качестве антенны использованы антенны типа MMDS-xx (xx – коэффициэнт усиления). Подходит антенна 2.4 ГГц, используемая для приема каналов "Космос-ТВ".

Вся первоначальная настройка точек доступа должна производится на земле, и только после проверки работоспособности на малом расстоянии, имеет смысл устанавливать оборудование на постоянное место.

Stargazer – открытая биллиговая система.

Система StarGazer предназначена для авторизации и учета трафика в локальных, домашних и офисных сетях. При разработке данной системы была поставлена цель создать продукт, который отвечал бы требованиям большинства локальных сетей для учета в них трафика и средств клиентов, а также безопасной авторизации клиентов.

Система построена по клиент-серверной технологии, что обеспечивает необходимую гибкость и быстродействие. В качестве сервера выступает машина с ОС Linux или FreeBSD, в качестве клиентов могут выступать машины как семейства Windows, так и клиенты с ОС Linux или FreeBSD. Так же клиентом может выступать любая ОС, в которой есть поддержка сетевых протоколов TCP/IP и веб-браузер. Система поддерживает подключаемые модули, что позволяет самостоятельно наращивать ее функционал.

Основные возможности системы:

- контроль над клиентами сети, их добавление, удаление, текущие корректировки;
- авторизация клиента, с последующим разрешением или запретом доступа в Internet;
- подсчет трафика по предварительно заданным направлениям и правилам;
- подсчет израсходованных клиентом средств и автоматическое отключение в случае их полного расходования;
- хранение дополнительной информации о клиенте, такой как домашний адрес, телефон и т.д.;
- автоматический пинг всех клиентов сети и вывод результатов на экран;
- ведение кредитной истории для всех клиентов;
- оперативное предоставление клиенту информации о его трафике и наличии средств;
- формирование подробных отчетов о состоянии клиентов;
- внешние модули.

Мониторинг сети

Для мониторинга может быть использован следующий набор утилит:

Iptraf – консольная утилита для сбора сетевой статистики. Она позволяет просматривать счетчики TCP-соединений как в байтах так и в пакетах, показывает статистику по интерфейсам, рабочей станции и индикаторы активности. Есть возможность устанавливать фильтры (отбор нужного трафика).

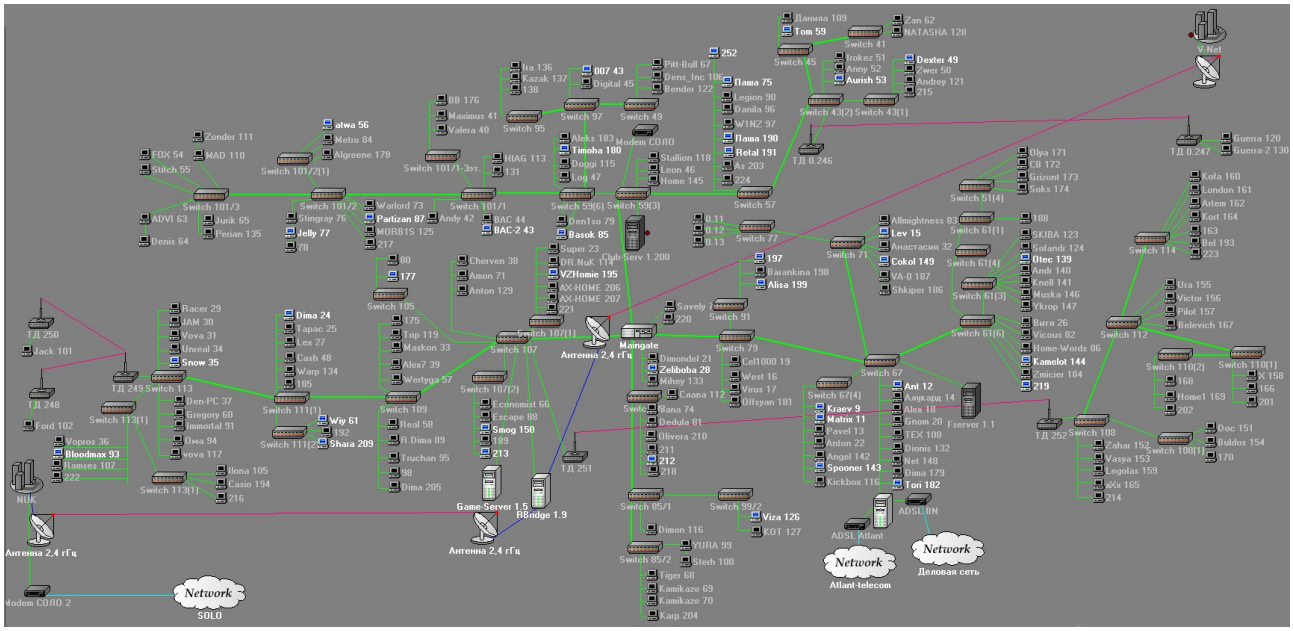
Iftop - аналог top по части использования сети: слушает трафик на указанном интерфейсе и отображает таблицу текущего использования по парам хостов. Удобно для ответа на вопрос "почему наш ADSL так тормозит?".

Arpwatch – трэкинг IP, MAC адресов в вашей сети. Регистрирует и отправляет по почте все изменения касающиеся хостов и их адресов. Сообщает о появлении новых хостов. Очень полезная утилита в на неуправляемых свичах.

Tcpdump –консольное средство мониторинга на уровне пакетов. Отображает полную информацию о составе, пути и порта назначения и отправки пакета. Если требуется разобрать ваш трафик до косточек, tcpdump ваш выбор (если у вас только консоль).

Wireshark (ранее Ethereal) – графический анализатор сетевых протоколов, который позволяет вам фиксировать и в интерактивном режиме просматривать содержание сетевых фреймов. Цель проекта состоит в том, чтобы создать качественный анализатор пакетов для Unix систем. Читает файлы данных tcpdump, Sniffer Pro, NetXray, MS Network Monitor, Novell's LanAlyzer и т.п. Поддерживает DNS, FDDI, FTP, HTTP, ICQ, IPV6, IPX, IRC, MAPI, MOUNT, NETBIOS, NFS, NNTP, POP, PPP, TCP, TELNET, X25 и т.д. Wireshark - это лучший (imho) инструмент отладки и мониторинга сети для графической оболочки.

Схема сети:



Использование математической программы Scilab в учебном процессе в БРУ и МГУ им. А.А. Кулешова

Виктор Журавлев - Белорусско-Российский университет - Могилев, Беларусь
Евгений Котяшев - Могилевский государственный университет им. А.А. Кулешова - Могилев, Беларусь

Излагается опыт использования математической программы Scilab в учебном процессе на кафедре "Физические методы контроля" Белорусско-Российского университета в дипломном проектировании и подготовке магистерской диссертации, а также на кафедре "Экспериментальная и теоретическая физика" МГУ им. А.А. Кулешова при разработке лабораторных работ и работе над кандидатской диссертацией. Дается оценка такого использования совместно с другим открытым программным обеспечением.

Введение

Современное состояние рынка универсального математического программного обеспечения позволяет говорить о практически полном доминировании проприетарных программ, таких как Matlab, Maple, Mathcad, Mathematica.

Современное положение в учебном процессе и учебных программах в БРУ и МГУ (каф. ФМК и ЭТФ)

В программе обучения на кафедре ФМК в виду технической направленности кафедры практически отсутствует комплексное обучение работе с математическим ПО, ограничиваясь несколькими часами лекций по Mathcad. Упор делается на преподавание навыков работы с системами автоматического проектирования (САПР). Поэтому здесь существует большой выбор того, что использовать во время обучения студентам, которые не связываются каким-либо отдельным продуктом, руководствуясь при этом больше общепринятостью и простотой его использования.

На кафедре ЭТФ читаются лекции по Mathcad и Maple, также проводятся факультативные занятия по работе с этим программами.

Нужно отметить, что на обеих кафедрах наблюдается уклон к использованию преимущественно Mathcad в большинстве случаев.

Перспектива использования Scilab в учебном процессе в БРУ и МГУ

Использование Scilab, впрочем как и любого открытого ПО, носит точечный и скорее единичный характер как на обеих кафедрах, так и в университетах в целом. Можно констатировать, что открытое ПО, включая ОС Linux, используют в своей работе единицы. Однако имеется перспектива такого использования, обусловленная тем, что открытое ПО, и в частности Scilab, для требований процесса обучения не уступает, а в большинстве случаев сравнимо по функциональности с проприетарным ПО.

Однако существует необходимость в разработке программ учебных курсов, касающихся работы со Scilab, понятных студентам и преподавателям и имеющих качественный уровень, не уступающий пособиям и материалам по работе с Mathcad и Maple.

Опыт использования Scilab в учебном процессе в БРУ и МГУ

На кафедре ФМК одним из авторов в дипломном проектировании (на тему "Влагомер для контроля влажности пиломатериалов") было целиком использовано отрытое ПО. Важно отметить, что программа Scilab оправдала себя широкими возможностями по обработке математических выражений, по построению диаграмм и графиков, а также по реализации

функций экспорта изображений. При работе над магистерской диссертацией Scilab оказалась способной обрабатывать широкий спектр экспериментальных данных.

На кафедре ЭТФ Scilab была использована при разработке лабораторных работ. В частности для математической обработки графических изображений профиля интенсивности в пучке лазера. Также существует задумка использовать Scilab в лабораторной работе с дозиметром "Сосна".

При работе над кандидатской диссертацией была сделана попытка использовать Scilab вместо Mathcad. Нужно отметить, что этот опыт увенчался успехом лишь отчасти (ввиду более углубленных теоретических изысканий и сложности расчетов). Наиболее важным преимуществом Scilab оказалась возможность посмотреть (благодаря открытому исходному коду) алгоритмы реализации используемых функций. К минусам же стоит отнести некоторую нестабильность работы программы Scilab, заключающуюся в вылетах из программы при определенных условиях и сбоях (малый размер кэша и пр.). Следует отметить также, что визуальный подход Mathcad к вводу формул заметно облегчает проверку корректности сложных математических выражений.

Выводы

На данный момент можно говорить только об использовании свободного ПО в университетах при личной инициативе, если руководство и/или программа обучения не запрещают и не ограничивают такое использование.

В целом пока в обоих университетах отсутствует какая-либо официальная политика использования свободных программ.

Вместе с тем практика доказала полную осуществимость использования в учебном процессе открытого ПО и в частности программы Scilab.

Также стоит отметить успехи в использовании Scilab в некоторых вузах (Донецкий национальный технический университет, Донецк, http://donntu.edu.ua) и написании хороших методических указаний на русском языке (http://www.scilab.land.ru/).

Mobile software development house

Ongan Mordeniz - Parise, France - IOOO BLRSoft

BLRSoft is a specialized software house in mobile embedded technologies. BLRSoft is 100% owned by French company ABAXIA, which is today recognized in Europe as the software house oriented UI (User Interface) and UX (User Experience) specialist. BLRSoft has in its strategy to innovate in this vision new mobile services and products with the support of the Open Source Code community.

Business overview

Through its market-leading active idle screen product, Abaxia's technology has a long history in helping service providers increase ARPU by putting services at a zero-click distance to the user and 'pushing' services directly to the front screen. Abaxia's embedded solutions enable service providers to introduce a consistent interface across major handset manufacturers; the company has secured handset OEM deals for large-scale distribution and deals with SIM & memory card manufacturers for just-in-time service delivery at the point of sale.

Abaxia's 12+ million software deployments, across 8 countries with tier-1 network operators, and 9 licensing agreements with major handset manufacturers afford the company an enviable target addressable market for service distribution, discovery and access. Moving forward, Abaxia is optimally positioned to help service providers to drive not only data ARPU, but more importantly recover the failing voice ARPU and secure advertising ARPU.

The main objective of BLRsoft is to develop and deploy mobile software application to vendor's handsets. ABAXIA and BLRSoft work very close with different vendors' terminal based on different technologies. Windows Mobile and Symbian based Oss like S60 and UIQ are mostly used for development purposes. With the upcoming Open Linux platform, ALP from Access, this will count to three and expanding BLRSoft innovation to more phones.

Product & Services

Abaxia's Mobile Desktop is a three-part suite of embedded applications which 'activate' the handset idle screen:

- Mobile Portal, a white-label, active idle screen application sold to tier-1 network operators and handset OEMs.
- Mobile Finder, an application that allows the user to swiftly locate handset features, operator services and user-generated content through predictive keyword matching, reducing click-distance from 10+ clicks to 3 clicks.
- Smart Agent, which allows distribution of mobile applications through the SIM and memory card, decoupling the handset customization cycle from the handset delivery cycle and allowing just-in-time customer segmentation.

With the Mobile Desktop suite, Abaxia's goal is to become the 'Google Desktop' of mobile, i.e. the front page for service search, access and promotion, across mobile handsets.

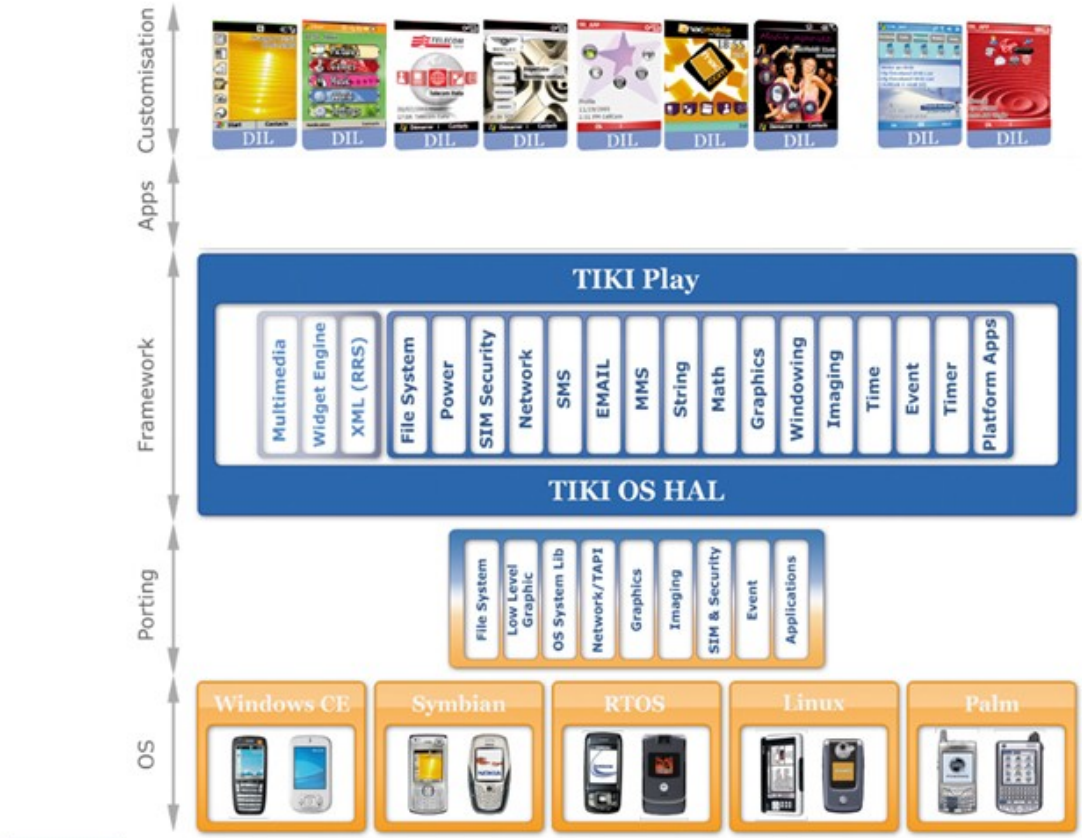
Technology Differentiation

Abaxia's and BLRSoft's handset-independent TIKI platform underpins all of its software applications and leads to:

- Reduced software development costs, from software re-use, which also leads to platform stability and robustness.
- Reduced time-to-market, as TIKI is optimized to enable quick and easy porting across platforms

• Drag-and-drop graphical UI development tool producing a single customization file for use across all handsets.
• Long-term know-how for in-ROM handset process deployment and access to private handset APIs.
• A “future-proofed", C-based platform, for highly-scalable RTOS porting to the largest number of handsets.

Architecture TIKI



Some customers and partners